Packet Scheduling in Switches with Target Outflow Profiles

Aditya Dua
Qualcomm Inc.
3165 Kifer Rd., Santa Clara, CA 95051
adua@qualcomm.com
Nicholas Bambos
Dept. of Elec. Engg. and Dept. of Mgmt. Sci. and Engg.
Stanford University
350 Serra Mall, Stanford, CA 94305
bambos@stanford.edu

Abstract

The problem of packet scheduling for traffic streams with target outflow profiles traversing input queued switches is formulated in this paper. Target outflow profiles specify the desirable inter-departure times of packets leaving the switch from each traffic stream. The goal of the switch scheduler is to dynamically select service configurations of the switch, so that actual outflow streams ("pulled" through the switch) adhere to their desired target profiles as accurately as possible.

Dynamic service controls (schedules) are developed to minimize deviation of actual outflow streams from their targets and suppress stream "distortion". Using appropriately selected subsets of service configurations of the switch, efficient schedules are designed, which deliver high performance at relatively low complexity. Some of these schedules are provably shown to achieve 100% pull-throughput. Moreover, simulations demonstrate that for even substantial contention of streams through the switch, due to stringent/intense target outflow profiles, the proposed schedules achieve closely their target profiles and suppress stream distortion.

The switch model investigated here deviates from the classical switching paradigm. In the latter, the goal of packet scheduling is primarily to "push" as much traffic load through the switch as possible, while controlling delay to traverse the switch and keeping congestion/backlogs from exploding. In the model presented here, however, the goal of packet scheduling is to "pull" traffic streams through the switch, maintaining desirable (target) outflow profiles.

Index Terms

Packet switching, Real-time Scheduling, Quality of Service, Dynamic Programming, Lyapunov Techniques.

I. Introduction

Real-time services such as multimedia streaming, video on demand, video telephony etc. continue to gain popularity amongst Internet users. These applications have strict quality-of-service (QoS) requirements with regard to packet delivery times and jitter. Scheduling algorithms employed in packet switches/routers play a key role in QoS provisioning for real-time Internet applications.

While early research on packet switching focused on the output-queued (OQ) switch architecture [1], [2], input-queued (IQ) switches have received much attention in recent times, owing to their scalable architecture. However, non-trivial scheduling/arbitration algorithms are needed to resolve contention between input traffic streams to ensure efficient operation of an IQ switch. Most research on IQ switch scheduling has revolved around performance metrics like throughput and average delay, which are conceived on *macro* time-scales (at the mean flow level). Numerous scheduling algorithms based on maximum weight matching (MWM), projective cone schedules (PCS), etc. have been proposed in the literature [3]- [6], all of which provably guarantee 100% (push)-throughput, with varying degrees of average delay performance. This body of literature, while important in its own right, does not address the problem of QoS provisioning for time/jitter sensitive real-time traffic, which entails performance engineering and control of the switch on *micro* time-scales (at the packet level).

In an initial effort to address the latter problem, in this paper, we develop IQ switch scheduling algorithms for traffic streams associated with *target outflow profiles*. The target profile of a traffic stream specifies the desirable (hence, the term "target") packet *inter-departure times* (IDT) of packets leaving the switch. In other words, the target outflow profile determines the *ideal* packet inter-departure times.

In the absence of congestion, packets from each stream will depart the switch in accordance with the associated target profile. However, contention between competing traffic streams for the shared switch fabric causes congestion in the switch. Consequently, the actual departure process of a stream *deviates* from the ideal departure process (as dictated by its target outflow profile). In other words, the stream outflow gets *distorted* by the switch, vis-à-vis its target profile. Thus, the objective of the switch service scheduler is to minimize the aggregate distortion of the target output profiles of all streams traversing the switch. That is, the scheduler must select switch *service traces* (sequences of switch configurations) such that the actual departure/outflow profiles of streams track their corresponding target profiles as accurately as possible. We call this the **Service Trace Control** (STC) problem for an IQ switch.

The motivation behind seeking a solution to the STC problem is to render packet switched networks somewhat "transparent" to timing/jitter sensitive multimedia traffic. The target outflow profiles are determined by the times at which consecutive packets need to be delivered to end users to ensure uninterrupted multimedia playout (the playout profile). High quality multimedia experience is provided to end-users if traffic streams negotiate routers/switches with minimal distortion. Note that the term "distortion" is simply used in this paper in connection to deviation of packet inter-departure times from their target profiles. The term is *not* used as in information theory and coding theory, where it has a deeper meaning. (deviation from target profiles).

In our switch model, delayed packets are not dropped, but instead are penalized for violating their target packet inter-departure times (IDT). The switch is also penalized for being ahead of the target packet IDTs. This is done to prevent buffer overflows at downstream nodes (flow control) and the end-user, as well to avoid starvation of best-effort traffic (i.e. without target outflow profiles) being served by the switch. This model is representative of half-duplex applications like lossless multimedia streaming (e.g. an online baseball game), where the end-user would much rather wait for a delayed packet than miss viewing the media content encoded in the delayed packet (which would happen if the switch drops delayed packets).

In our framework, packets can be thought of as being associated with *soft deadlines* for their inter-departure times (IDT). Any positive deviation (exceeding the deadline) from the target IDTs manifests itself as a soft deadline violation, which carries a penalty/cost. The "softness" of a deadline is reflected by the cost associated with its violation (the lower the violation cost, the softer the deadline). On the other hand, any negative deviation from the target IDTs (transmitting before a target inter-departure time) is also a soft deadline violation, and carries a cost (e.g. for stressing downstream receivers with potential buffer overflows). The service trace control (STC) problem thus translates to minimization of aggregate soft deadline violation cost over all traffic streams. This is explained in detail in Section II.

In the classical packet switching paradigm (see [3]- [6])) incoming traffic flows compete for switch service. The scheduler's objective is to control the congestion buildup (and avoid excessive backlogs), given the traffic load. Alternatively, the scheduler tries to maximize the inflow load that can be "pushed" through the switch, without the packet backlogs exploding. Hence, it tries to maximize the "push-throughput". In the switch model studied in this paper, the issue is very different. Packets streams are "pulled" through and out of the switch. The packets initially reside in input queues, organized as virtual output queues (VOQ). Recall that the scheduler's objective is now to pull the streams through and out of the switch, so that their outflow packet inter-departure times (IDT) deviate as little as possible from specified targets and the outflow stream distortion is minimized. But if the target IDTs are too short (outflow target profiles have high intensity) the switch may not be able to keep up and the distortion of one or more streams may grow excessively over time. Thus, the scheduler can now be viewed as trying to maximize the "pull-throughput" of the switch, i.e., supply the most intense outflow streams, while keeping their distortions under control. This is explained in detail in Sections II and V.

A. Related work

The case of scheduling periodic messages through IQ switches has been addressed in the literature. In that case, packets for each traffic stream are generated periodically, and the maximum time allowed for transmission of a

packet is equal to the period of the stream. A schedule is deemed feasible if all messages meet their deadline requirements. Note that the periodic model is a special case of our general model, with constant inter-departure times (equal to the period of the stream). Inukai [7] showed that a feasible schedule can be constructed when the periods of all streams are equal and both input and output link utilization are less than 1. Liu et al. [8] conjectured that Inukai's conclusion holds for traffic streams with arbitrary periods and also proposed heuristic scheduling algorithms based on the earliest deadline first (EDF) and minimum laxity first (MLF) policies. The performance of their heuristics degrades rapidly with switch size. In support of the conjecture, Giles et al. [9] proposed the nested periodic scheduling (NPS) rule, which finds a feasible schedule when each period divides all longer periods and link utilization is less than 1. NPS also finds a feasible schedule for arbitrary message periods, provided the link utilization is no more than 1/4. The computational complexity of NPS is $\mathcal{O}(N^4)$ for an $N \times N$ switch. Rai et al. [10] developed heuristic weighted round robin (WRR) scheduling policies for multiclass periodic traffic, with an online implementation complexity of $\mathcal{O}(N^3)$. More recently, Lee et al. [11] proposed the Flowbased Iterative Packet Scheduling (FIPS) algorithm for periodic traffic with two classes, which minimizes the number of dropped packets when the switch is overloaded. They extended the FIPS algorithm to design efficient heuristics for arbitrary multiclass traffic. Their proposed algorithms outperform MLF and EDF based policies, but have a complexity of $\mathcal{O}(N^{4.5})$.

On a different strand of research, Li et al. [12] developed a frame-based scheduler with guaranteed delay and jitter bounds for leaky-bucket constrained traffic. Chang et al. proposed schemes for providing delay guarantees in IQ switches based on the Birkhoff-von Neumann (BV) decomposition of the input rate matrix in [13] and based on EDF for load balanced switches (see [14]) in [15]. Their schemes have an offline computational complexity of $\mathcal{O}(N^{4.5})$ and an online memory requirement of $\mathcal{O}(N^3 \log N)$. Keslassy et al. [16] proposed a frame based scheduler based on the BV decomposition to guarantee low jitter, under the assumption that jitter sensitive traffic forms a small fraction of the overall switch load.

A common feature of the above works is that they deal with scheduling of smooth/regular traffic (completely characterized by a single fixed rate known to the scheduler). However, traffic arriving to a switch can be irregular due to the bursty nature of traffic sources (e.g. variable bit rate video), due to flow aggregation, or due to jitter induced by upstream switches. Further, rates of different streams are not always known to the scheduler. Also, these schemes have significant computational complexity, making them relatively difficult to implement in high speed switches.

For completeness, we also mention two other somewhat relevant bodies of work, akin in spirit to our modeling approach. Our "soft deadline" point of view discussed before is reminiscent of the time/utility function (TUF) approach introduced by Jensen et. al. [17] to study scheduling in real-time operating systems. Moreover, our notion of target profiles for different traffic streams is reminiscent of the rich set of network calculus tools developed by Cruz ([18] and several subsequent works with others) to study the problem of providing deterministic QoS guarantees in time-slotted virtual circuit networks, based on the notion of service curves.

B. Contributions

The key contributions of our work are two-fold. Firstly, we develop a novel *outflow aware* switching framework, based on the idea of shaping the switch outflow streams to match desired/target profiles. While we exclusively study this model in the context of an IQ switch, the core ideas are more widely applicable to any queuing system where competing users/jobs are associated with inter-departure time (IDT) constraints.

Secondly, we develop relatively low complexity scheduling policies for IQ switches, using the idea of switch configuration subset based schedules. The idea is to partition the huge set of possible switch service configurations (of size $N!$) into smaller subsets of size N each, and schedule the switch using only one subset in every time-slot. The resulting policies achieve relatively low complexity. The results presented here provide a substantial extension of the research thread initiated in [20], [21], where some early observations regarding the studied switch model were made.

In contrast to the previously cited works, in our switch model we do not make any assumptions on the rate, periodicity etc. of traffic streams traversing the switch. We also develop a family of scheduling policies achieving lower complexity of $\mathcal{O}(N^2)$ per time-slot, which could be manageable from an implementation point of view in certain practical situations.

C. Organization of the paper

The remainder of this paper is organized as follows: In Section II, we first formulate the service trace control (STC) problem for minimizing stream distortion with respect to their target profiles as a finite-horizon dynamic program [22]. We then establish the optimality of a *greedy policy* for a 2×2 switch and explore its feasibility as a heuristic policy for bigger switches. Subsequently, we introduce the notion of switch configuration subset based STC in Section III. In Section IV, we develop the notion of *meta-queues*, which yields an alternative view of subset based STC and also provides a general framework for designing different families of STC policies. In Section V, we define the admissible region of the switch and show (using Lyapunov techniques) that subset based STCs, with appropriate subset selection rules, guarantee finite deviation from targets for all traffic streams, under *any* admissible load. Experimental evaluation of various proposed scheduling/STC policies in Section VI demonstrates high-performance under various stress regimes. The paper concludes in Section VII.

D. Notations and conventions

Notations and conventions employed throughout the paper are summarized here for convenience. All vectors and sequences are denoted in **boldface**. For a vector $\mathbf{x}$, the n^{th} element is denoted by x_n, and for a vector $\mathbf{x}_i$, the n^{th} element is denoted by $x_{i,n}$. $\mathbb{N}$ denotes the set of natural numbers, $\mathbb{Z}$ denotes the set of integers, and $\mathbb{Z}_+$ denotes the set of non-negative integers. $\mathbf{0}$ denotes the all zeros vector and $\mathbf{1}$ denotes the all ones vector. $\mathbf{e}_i$ denotes the i^{th} unit vector in $\mathbb{R}^N$, i.e., a vector with a 1 in the i^{th} location and 0's elsewhere. Further, $\mathbf{e}_0 = \mathbf{0}$. The inner product between two vectors $\mathbf{x}$ and $\mathbf{y}$ is denoted $\langle \mathbf{x}, \mathbf{y} \rangle$. Finally, the "big-oh" notation $f(N) = \mathcal{O}(g(N))$ is used to indicate that $\exists\, c > 0$ such that $f(N) \leq cg(N)$ for large enough N.

II. Minimizing Stream Distortion

A. Switching model

Consider an input queued (IQ) switch with virtual output queues (VOQs) at all input ports to prevent head-of-line (HOL) blocking. There are N^2 VOQs in an $N \times N$ switch with N input and N output ports, as shown in Fig. 1. Both input and output ports are indexed $1, \ldots, N$. The i^{th} VOQ stores packets destined from input port $\lfloor (i-1)/N \rfloor + 1$ to output port $(i-1) \mod N + 1$ and is denoted $\mathcal{Q}_i$. The switch operates in slotted time. Every input (output) port can be connected to at most one output (input) port in a time-slot. An $N \times N$ switch can be set into $N!$ possible *configurations*. Each configuration is associated with a unique *configuration vector* of length N^2. Let $\mathbf{v}_i = (v_{i,1}\ v_{i,2} \ldots v_{i,N^2}) \in \mathcal{V}$ denote the i^{th} configuration vector, where $\mathcal{V}$ is the set of all possible configuration vectors. Then, $v_{i,j} = 1$ if $\mathcal{Q}_j$ is served when the switch is set in configuration $\mathbf{v}_i$ and $v_{i,j} = 0$ otherwise. We use the terms configuration and configuration vector interchangeably throughout the paper.

Example 1: Two possible configuration vectors for a 2×2 switch are $\mathbf{v}_1 = (1\ 0\ 0\ 1)$ and $\mathbf{v}_2 = (0\ 1\ 1\ 0)$. If a 2×2 switch is configured with configuration vector $\mathbf{v}_1$, the first (second) input port is connected to the first (second) output port. If the switch is configured with vector $\mathbf{v}_2$, the first (second) input port is connected to the second (first) output port.

In each time-slot, a single cell can be transferred from an input port to an output port, if those ports are connected in the selected switch configuration. This cell/packet resides in the VOQ associated with the input-output port pair. We use the terms packet and cell interchangeably. Indeed, a cell is a packet of size 1. The underlying assumption is that a packet of size K cells can be "broken" into K cells for individual processing, and reassembled at the output of the switch.

We assume there is a large (theoretically infinite) supply of cells/packets residing at the VOQs initially, so that VOQs never run out of packets. For example, one may consider a switch in a video server farm, where video content is retrieved from hard disks and streamed via the switch to remote users. The switch VOQs are directly fed with video packets from the server disk and never (rarely) empty until the streamed content transmission completes. Analogous scenarios emerge in storage area network switches, where large files are retrieved from hard disks and streamed via switches to users.

Every VOQ is associated with a *traffic stream*, characterized by a **Target Stream Profile** (TSP). The traffic stream's cells/packets are stored in the associated VOQ. The TSP is the *desirable profile of outflow traffic*, i.e., of the stream leaving the switch. It basically specifies the time-slots in which cells of the stream should ideally depart

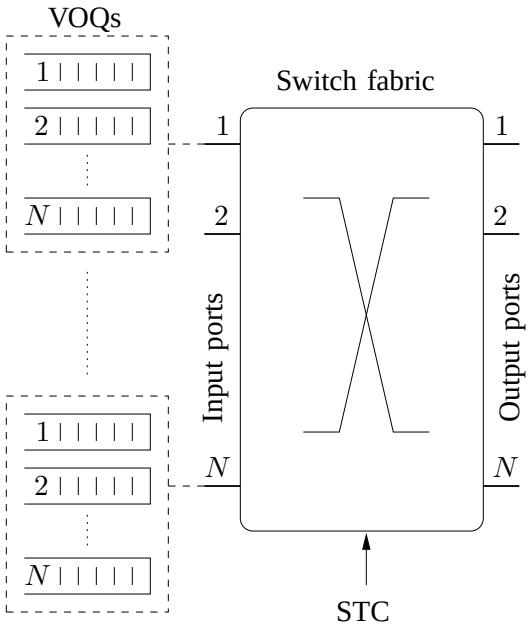

Fig. 1. Input Queued Switch

the switch. Alternatively, it characterizes the ideal time distance (number of slots) for releasing two consecutive cells from the stream's VOQ and getting them through and out of the switch.

Technically, the TSP is a sequence of "0"s and "1"s which specifies the packet *inter-departure time* (IDT) targets/constraints between packets in the stream. Let

$$\mathbf{s} = (s^1, s^2, \ldots) \tag{1}$$

denote the TSP for a typical traffic stream. Suppose that the k^{th} "1" in $\mathbf{s}$ occurs at location $\tau \in \mathbb{N}$ and the $(k+1)^{st}$ "1" occurs at location $\tau + \delta_k$, for some $\delta_k \in \mathbb{N}$. The interpretation is that the k^{th} packet in the stream should ideally depart the switch in the τ^{th} time-slot, the $(k+1)^{st}$ packet in the stream should depart the switch in the $(\tau+1)^{st}$ time-slot, and therefore the desired inter-departure time (IDT) target between the k^{th} and $(k+1)^{st}$ packets of the stream is $(\tau + \delta_k) - \tau = \delta_k$ time-slots. From the TSP we derive the **cumulative Target Stream Profile** (cTSP), denoted $\mathbf{S} = (S^1, S^2, \ldots)$, $S^t \in \mathbb{Z}_+ \; \forall \; t$, where

$$S^t \triangleq \sum_{\tau=1}^{t} s^\tau, \; t = 1, 2, \ldots \tag{2}$$

is the number of packets of the stream which should ideally have departed the switch by the end of the t^{th} time-slot.

Example 2: We illustrate the concepts of TSP and cTSP through an example. Let the TSP of a stream be given by $\mathbf{s} = (0, 1, 0, 0, 1, 0, 1, \ldots)$. This implies that the 1^{st} packet of the stream should ideally depart the switch in the 2^{nd} time-slot, the 2^{nd} packet should ideally depart in the 5^{th} time-slot, the 3^{rd} packet should ideally depart in the 7^{th} time-slot, and so on. The entries of the cTSP are computed (by definition) as $S^1 = s^1, S^2 = s^1 + s^2, \ldots$. Thus, we have $\mathbf{S} = (0, 1, 1, 1, 2, 2, 3, \ldots)$. The interpretation is that no packets from this stream should have departed the switch by the end of the 1^{st} time-slot, exactly one packet should have departed by the end of the 2^{nd} time-slot, etc.

Example 3: A special example is that of **periodic traffic** (of period δ) with fixed inter-departure times, i.e., $\delta_k = \delta \; \forall \; k$. In this case, $s^t = 1$ if $t \mod \delta = 0$, and $s^t = 0$ otherwise. Further, $S^t = \lfloor t/\delta \rfloor$. In general, the IDT targets may not be constant but vary substantially, for instance, because of coding dependencies of cell/packets in video streams, etc.

To characterize the service provided by the switch, we associate with every stream a **Received Service Trace** (RST), also a sequence of "0"s and "1"s. This is the actual (not desired) service sequence received by the stream. Let

$$\mathbf{r} = (r^1, r^2, \ldots) \tag{3}$$

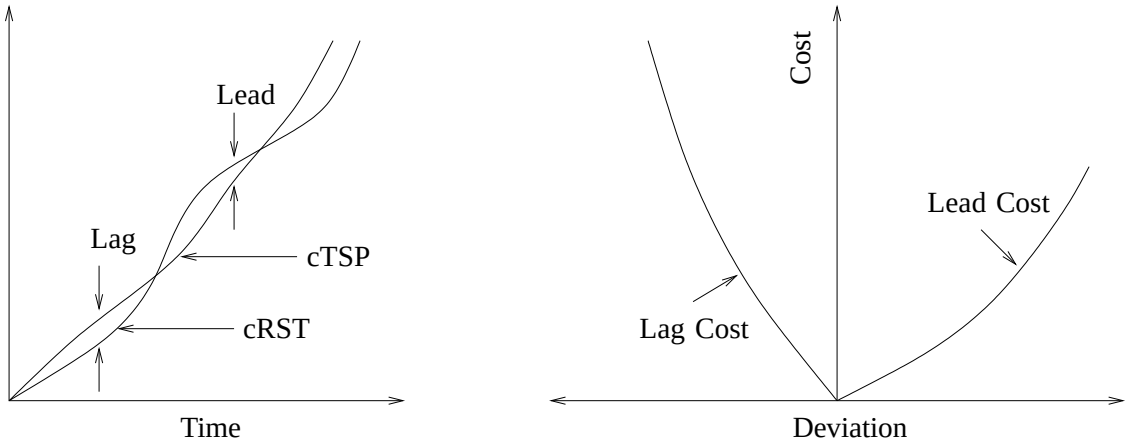

Fig. 2. The left side illustrates the notions of cTSP, cRST and deviation. The right side depicts a typical deviation cost function.

denote the RST associated with a typical stream. Then, $r^\tau = 1$ if the switch serves a packet from the stream in the τ^{th} time-slot, and $r^\tau = 0$ otherwise. Similar to the cTSP, we derive the **Cumulative Received Service Trace** (cRST), denoted $\mathbf{R} \triangleq (R^1, R^2, \ldots)$, $R^t \in \mathbb{Z}_+ \; \forall \; t$, where

$$R^t = \sum_{\tau=1}^{t} r^\tau, \; t = 1, 2, \ldots \tag{4}$$

is the number of packets of the stream which have actually departed the switch by the end of t^{th} time-slot.

Example 4: We illustrate the concepts of RST and cRST through an example. Consider a traffic stream with its TSP as given by Example 2. Now, suppose the RST for this stream is given by $\mathbf{r} = (0, 1, 0, 1, 0, 0, 0, 1, \ldots)$. The interpretation is that the 1^{st} packet of the stream departed the switch in the 2^{nd} time-slot, the 2^{nd} packet departed in the 4^{th} time-slot, the 3^{rd} packet departed in the 8^{th} time-slot, and so on. By definition, the cRST is constructed as $R^1 = r^1, R^2 = r^1 + r^2, \ldots$, yielding $\mathbf{R} = (0, 1, 1, 2, 2, 2, 2, 3, \ldots)$. The interpretation is that no packets from the stream were released from the switch by the end of the 1^{st} time-slot, one packet was released by the end of the 2^{nd} time-slot, etc.

Ideally, for every stream we desire $R^t = S^t \; \forall \; t$, which implies that every stream traverses the switch without experiencing any "distortion" of its target profile. However, this goal is not always realizable due to congestion caused by contention between competing streams for the shared switch fabric. If for a particular stream $R^t > S^t$, the stream has received more service than it requires to satisfy its packet inter-departure time (IDT) constraints and is said to be **leading** at time t. If $R^t < S^t$, the stream has received less than its desired amount of service and is said to be **lagging** at time t. To quantify distortion of target profiles due to congestion, we track for every traffic stream its **deviation**, denoted $\mathbf{d} \triangleq (d^1, d^2, \ldots)$, $d^t \in \mathbb{Z} \; \forall \; t$, where

$$d^t \triangleq R^t - S^t, \; t = 1, 2, \ldots \tag{5}$$

which quantifies the excess or deficiency in service catered to the stream by the switch as a function of time. A negative deviation (**lag**) indicates missed deadlines and is undesirable from a QoS provisioning perspective. A positive deviation (**lead**) is undesirable because it can cause buffer overflows at downstream switches and the end user and lead to starvation of delay tolerant flows traversing the switch.

Example 5: We illustrate the notion of deviation through an example. Consider a traffic stream with its TSP as given by Example 2 and RST as given by Example 4. Recall that for this stream, the cTSP is given by $\mathbf{S} = (0, 1, 1, 1, 2, 2, 3, \ldots)$ and the cRST is given by $\mathbf{R} = (0, 1, 1, 2, 2, 2, 2, 3, \ldots)$. Taking an elementwise difference, the deviation is given by $\mathbf{d} = \mathbf{R} - \mathbf{S} = (0, 0, 0, 1, 0, 0, -1, \ldots)$. Note that the 1^{st} packet of the stream gets served by the switch on time, the 2^{nd} packet gets served one time-slot in advance, and the 3^{rd} packet gets served one time-slot later than desired. The stream is therefore "leading" for one time-slot immediately after the departure of the 2^{nd} packet, and is lagging in the 7^{th} time-slot, which is the desired/target departure time of the 3^{rd} packet in the stream.

The ideas introduced in this section are depicted in Fig. 2. While the cTSP and cRST curves are shown to be "smooth" in the figure for illustration, note that for the discrete-time model studied in this paper (at most one cell per VOQ processed by the switch in a time-slot), the curves will look like staircase functions, with the step size equal to 1.

B. Finite horizon dynamic programming (DP) formulation

Consider a finite horizon of $T > 0$ time-slots indexed by $t \in \{1, \ldots, T\}$. Let $\mathbf{s}_i = (s_i^1, \ldots, s_i^T)$ and $\mathbf{S}_i = (S_i^1, \ldots, S_i^T)$ denote the the first T entries of the TSP and cTSP of VOQ $\mathcal{Q}_i$, respectively. Define

$$\mathbf{x}^t \triangleq (s_1^t, \ldots, s_{N^2}^t),\ X^t \triangleq (S_1^t, \ldots, S_{N^2}^t). \tag{6}$$

Thus, $\mathbf{x}^t$ ($\mathbf{X}^t$) is a vector of the t^{th} entries of the TSP (cTSP) of all the N^2 streams. We shift from the $\mathbf{s}_i$ (S_i) notation to the x^t (X^t) one in order to change the point of view from being focused on each individual queue/stream i to tracking (all queues/streams at) each time-slot $t \leq T$. To clarify the notation further, consider a $T \times N^2$ matrix with $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^t$ as its rows. The i^{th} column of this matrix comprises of the TSP entries for $\mathcal{Q}_i$ over the time horizon of interest, viz. $\{1, \ldots, T\}$, In matrix terminology, the i^{th} column is the transpose of the TSP of $\mathcal{Q}_i$. On the other hand, the t^{th} row comprises of the t^{th} entries of all N^2 traffic streams traversing the switch. Next, define

$$\mathbf{d}^t \triangleq (d_1^t, \ldots, d_{N^2}^t) \tag{7}$$

as the **state** of the switch in the t^{th} time-slot, where d_i^t is the *deviation* (as defined in (5)) of the stream associated with VOQ $\mathcal{Q}_i$ in the t^{th} time-slot.

Since deviations from target profiles are undesirable, they are associated with a "cost". In particular, to the i^{th} stream we assign the cost function $\phi_i(k)$, which reflects the cost associated with a deviation $k \in \mathbb{Z}$. We assume the following:

1) $\phi_i(0) = 0$ (zero deviation is desirable)
2) $\phi_i(k)$ is non-negative and increasing for both $k > 0$ and $k < 0$ (since both leads and lags are undesirable)
3) $\phi_i(k)$ is convex (the cost associated with deviation increases at a positive rate as the deviation increases in magnitude)

A sample cost function which satisfies the above properties is depicted on the right side of Fig. 2. An example of a cost function which we will often use in this paper is the quadratic cost function $\phi_i(k) = k^2$. Finally, let

$$\Phi(\mathbf{d}^t) \triangleq \sum_{i=1}^{N^2} \phi_i(d_i^t) \tag{8}$$

denote the sum of the deviation costs of all VOQs.

Remark 1: It is important to note that unlike the packet inter-departure time constraints, the cost functions are not an inherent part of the problem, but are instead extraneously assigned by the switch controller for the purpose of service trace control. Thus, the switch controller has the freedom to *tune* these cost functions in order to optimize switch performance.

Remark 2: In our modeling framework, packets can be thought of as being associated with *soft deadlines*. For the more conventional case of strict deadlines, the "value" of a packet is constant prior to its deadline and zero thereafter. As a result, a packet is dropped if it has not departed the queue before its due date. In our context, where a typical motivating application is multimedia streaming, lossless delivery of packets is sought. The "value" of a packet reaches its peak at its target delivery time (as dictated by the TSP). The packet is treated as less valuable (but not dropped) if received either before or after its target time. In this sense, the "softness" of the deadline constraints for a traffic stream is quantified by the steepness of the associated cost function.

In every time-slot, the **Service Trace Controller** (STC) drives the evolution of service traces for various traffic streams by setting the switch in one of $N!$ possible configurations (chosen from the set $\mathcal{V}$) or idling the switch.

Definition 1: A **policy** $\Pi_T = \{\mathbf{v}^t \in \mathcal{V} \cup \{\mathbf{0}\},\ t = 1, \ldots, T\}$ is defined as a sequence of switch configurations selected by the service trace controller in time-slots $t = 1, \ldots, T$.

Given the initial state $\mathbf{d}^0$, we are interested in computing the optimal policy (one which minimizes the total cost over a finite horizon) $\Pi_T^\star$ which satisfies

$$\Pi_T^\star = \arg\min_{\Pi_T} \left\{ \sum_{t=1}^{T} \Phi(\mathbf{d}_{\Pi_T}^t) \right\}, \tag{9}$$

where $\mathbf{d}_{\Pi_T}^t$ denotes the state of the switch at the beginning of the t^{th} time-slot under policy Π_T. We will adopt the methodology of *dynamic programming* (DP) to compute $\Pi_T^\star$.

Suppose Π_T chooses configuration vector $\mathbf{v}^t = \mathbf{v} = (v_1, v_2, \ldots, v_{N^2})$ in the t^{th} time-slot. The deviation of the VOQ $\mathcal{Q}_i$ increases by 1 at the end of the t^{th} time-slot if it is served by configuration $\mathbf{v}$, i.e., $v_i = 1$. Also, the deviation of the $\mathcal{Q}_i$ decreases by 1 if its TSP has a non-zero entry in the t^{th} location, i.e., $x_i^t = 1$. Note that x_i^t (the i^{th} component of $\mathbf{x}^t$) is simply s_i^t, from (6). More compactly, the new deviation vector at the beginning of the $(t+1)^{st}$ time-slot is given by

$$\mathbf{d}_{\Pi_T}^{t+1} = \mathbf{d}_{\Pi_T}^t + \mathbf{v}^t - \mathbf{x}^t. \tag{10}$$

Let $V^t(\mathbf{d})$ be the cost incurred by $\Pi_T^\star$ over time-slots $t, \ldots, T$, starting in state $\mathbf{d}$ at the beginning of the t^{th} time-slot. In dynamic programming terminology, $V^t(\cdot)$ is referred to the as the *cost-to-go* function, and is recursively computed from the following DP equations for $t = 1, \ldots, T$

$$V^t(\mathbf{d}) = \min_{\mathbf{v} \in \mathcal{V} \cup \{\mathbf{0}\}} \left\{ \underbrace{V^{t+1}(\mathbf{d} + \mathbf{v} - \mathbf{x}^t)}_{\text{Cost-to-go in the next time-slot}} + \underbrace{\Phi(\mathbf{d} + \mathbf{v} - \mathbf{x}^t)}_{\text{Instantaneous cost}} \right\}, \tag{11}$$

and the boundary conditions $V^{T+1}(\mathbf{d}) = 0 \; \forall \; \mathbf{d}$. We will henceforth refer to $\mathbf{d} - \mathbf{x}^t$ as the **deviation vector**.

C. Myopic/Greedy service trace control

Observe from (11) that the optimal decision in state $\mathbf{d}$ in the t^{th} time-slot is determined by the cost-to-go in the $(t+1)^{st}$ time-slot, as well as the instantaneous cost. Now consider a *myopic policy*, which is "greedy" with respect to the instantaneous cost, i.e., ignores the cost-to-go in the next time-slot while making its current scheduling decision. In particular, the myopic policy chooses configuration $\mathbf{v}^\star$ in the t^{th} time-slot in state $\mathbf{d}$ such that

$$\mathbf{v}^\star = \arg\min_{\mathbf{v} \in \mathcal{V} \cup \{\mathbf{0}\}} \left\{ \Phi(\mathbf{d} + \mathbf{v} - \mathbf{x}^t) \right\}. \tag{12}$$

In general, the myopic policy need not be optimal. However, for the scheduling problem at hand, the myopic policy is provably optimal for the case $N = 2$. For $N = 3, 4$, numerical analysis reveals that the myopic policy is close to optimal. The cost of computing the optimal policy becomes prohibitive as N gets bigger ($N! + 1$ possible decisions need to be evaluated in every possible state of the switch over a period of T time-slots).

Theorem 1: The optimal finite horizon policy $\Pi_T^\star$ for a 2×2 switch ($N = 2$) is myopic.

Proof: See Appendix VIII-A. ∎

Example 6: For concreteness and as a key example, let us assign quadratic cost functions to all traffic streams, i.e., $\phi_i(k) = k^2 \; \forall \; i$. For any $\mathbf{v} \in \mathcal{V} \cup \{\mathbf{0}\}$ we have

$$\Phi(\mathbf{d} + \mathbf{v} - \mathbf{x}^t) = \underbrace{\langle \mathbf{d} - \mathbf{x}^t, \mathbf{d} - \mathbf{x}^t \rangle}_{\text{Policy independent}} + \underbrace{\langle \mathbf{v}, \mathbf{v} \rangle + 2\langle \mathbf{d} - \mathbf{x}^t, \mathbf{v} \rangle}_{\text{Policy dependent : } \varsigma(\mathbf{v})}. \tag{13}$$

The myopic policy in this case reduces to

$$\mathbf{v}^\star = \begin{cases} \widetilde{\mathbf{v}}^\star & ; \quad 2\langle \mathbf{d} - \mathbf{x}^t, \widetilde{\mathbf{v}}^\star \rangle + N \leq 0 \\ \mathbf{0} & ; \quad \text{else,} \end{cases} \tag{14}$$

where $\widetilde{\mathbf{v}}^\star \triangleq \arg\min_{\mathbf{v} \in \mathcal{V}} \left\{ \langle \mathbf{d} - \mathbf{x}^t, \mathbf{v} \rangle \right\}$.

The idea is as follows: We want to find a $\mathbf{v} \in \mathcal{V}$ which minimizes the policy dependent part. The set $\mathcal{V}$ is the set of all N^2 switch configuration plus the zero configuration (switch idle). For $\mathbf{v} = 0$, the policy dependent part is 0. For all non-zero configurations, $\langle \mathbf{v}, \mathbf{v} \rangle = N$, and the policy dependent part is $2\langle \mathbf{d} - \mathbf{x}^t, \mathbf{v} \rangle + N$. This term is minimized by $\widetilde{\mathbf{v}}^\star$ (by definition). Thus, we pick the $\mathbf{v}$ which minimizes the min of 0 and $2\langle \mathbf{d} - \mathbf{x}^t, \widetilde{\mathbf{v}}^\star \rangle$.

D. Partial configurations

We begin this section with a definition.

Definition 2: For an $N \times N$ IQ switch, a switch configuration $\mathbf{v} \in \mathcal{V}$ is called **complete** if $\langle \mathbf{v}, \mathbf{1} \rangle = N$ and is called **partial** if $\langle \mathbf{v}, \mathbf{1} \rangle \leq N$.

In other words, in a complete configuration, *every* input port is connected to an output port, while in a partial configuration, some of the input ports may be idle.

So far we have assumed that the service trace controller (STC) either selects a *complete* configuration (*every* input port is connected to an output port) or idles the switch. However, operating the switch using complete configurations only is not sufficient to exercise individual control on service traces of different streams, as illustrated by the next example.

Example 7: Consider a 2×2 switch where the streams for $\mathcal{Q}_1$ and $\mathcal{Q}_4$ are periodic with periods 2 and 4 respectively, and $\mathcal{Q}_2$ and $\mathcal{Q}_3$ are empty (no new arrivals). In our notation, this translates to $\mathbf{x}^1 = (0,0,0,0), \mathbf{x}^2 = (1,0,0,0), \mathbf{x}^3 = (0,0,0,0), \mathbf{x}^4 = (1,0,0,1)$ and $\mathbf{x}^t = \mathbf{x}^{t \mod 4} \; \forall \; t$. The myopic policy (which is optimal) given by (14) either selects $\mathbf{v}_1 = (1\ 0\ 0\ 1)$ or $\mathbf{0}$ in every time-slot. The configuration vector $\mathbf{v}_2$ is never selected because both queues serviced by $\mathbf{v}_2$ are empty. It is easily verified that either the lag of $\mathcal{Q}_1$ or the lead of $\mathcal{Q}_4$ grow without bound under the optimal policy.

To exercise individual control over service traces, we allow the STC to use *partial* configurations. Suppose complete configuration $\mathbf{v} \in \mathcal{V}$ serves VOQs indexed by the set $\mathcal{I} = \{i_1, \ldots, i_N\}$. Any partial configuration $\bar{\mathbf{v}}$ extracted from $\mathbf{v}$ is characterized by a vector $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_N)$, where $\xi_j = 1$ if $\bar{\mathbf{v}}$ serves $\mathcal{Q}_{i_j}$ and $\xi_j = 0$ if $\bar{\mathbf{v}}$ idles $\mathcal{Q}_{i_j}$. Thus, 2^N partial configurations can be extracted from any complete configuration.

Example 8: Consider configuration $\mathbf{v}_1 = (1\ 0\ 0\ 1)$ for a 2×2 switch. This configuration serves VOQs $\mathcal{Q}_1$ and $\mathcal{Q}_4$. The partial configuration set $\{(1\ 0\ 0\ 0), (0\ 0\ 0\ 1), \mathbf{0}, \mathbf{v}_1\}$ can be extracted from the complete configuration $\mathbf{v}_1$. The first partial configuration in the set corresponds to $\boldsymbol{\xi} = (1,0)$, the second partial configuration corresponds to $\boldsymbol{\xi} = (0,1)$, etc. Note that the configurations $\mathbf{0}$ and $\mathbf{v}$ are always part of the configuration set associated with complete configuration $\mathbf{v}$.

E. The Maximum Sum of Lags (MSL) policy

Let us revisit the myopic service trace control policy for the case of quadratic cost functions (Example 6), allowing for partial configurations this time. Recall from (13) that the policy dependent part in Φ is $\varsigma(\mathbf{v}) = \langle \mathbf{v}, \mathbf{v} \rangle + 2\langle \mathbf{d} - \mathbf{x}^t, \mathbf{v} \rangle$. If $\mathbf{v}$ serves VOQs indexed by set $\mathcal{I}$, $\varsigma(\mathbf{v})$ can be rewritten as $\sum_{j \in \mathcal{I}} v_j^2 + 2 \sum_{j \in \mathcal{I}} v_j (d_j - x_j^t)$. Since $\mathbf{v}$ is a complete configuration, $v_j = 1 \; \forall \; j \in \mathcal{I}$. Now, split $\mathcal{I}$ into two disjoint subsets, $\mathcal{I}_+$ and $\mathcal{I}_-$, where $\mathcal{I}_+ = \{j \in \mathcal{I} : d_j - x_j^t \geq 0\}$ and $\mathcal{I}_- = \{j \in \mathcal{I} : d_j - x_j^t < 0\}$. Note that $\mathcal{I}_+ \cup \mathcal{I}_- = \mathcal{I}$ and $\mathcal{I}_+ \cap \mathcal{I}_- = \emptyset$. Clearly, $\varsigma(\mathbf{v})$ can be strictly decreased by setting $v_j = 0 \; \forall \; j \in \mathcal{I}_+$. Doing so is equivalent to extracting a partial configuration from $\mathbf{v}$ by idling all VOQs with non-negative deviation. We therefore get the following two-step service trace control policy, which we refer to as the **Maximum Sum of Lags** (MSL) policy (see Table I).

1) Select $\widetilde{\mathbf{v}}^\star = \arg\min_{\mathbf{v} \in \mathcal{V}} \big\{ \langle \mathbf{d} - \mathbf{x}^t, \mathbf{v} \rangle \big\}$.
2) Extract a partial configuration from $\widetilde{\mathbf{v}}^\star$ by idling all VOQs with non-negative deviation.

The name of the policy arises from the fact that it selects the switch configuration whose associated VOQs have the largest sum "lag" (as defined in Section II-A).

The computational complexity of MSL is $\mathcal{O}(N^3)$ per time-slot, since Step 1 involves a maximum weight matching (MWM) computation on a bipartite graph [19]. Note that the edge weights used to compute this matching are in fact the deviations associated with the VOQs. Switching algorithms which use VOQ backlogs as the edge weights for computing MWM have been studied extensively in the literature, in the context of throughput maximizing switches (e.g. [3]). While $\mathcal{O}(N^3)$ complexity is a significant improvement over the optimal policy, algorithms to compute the maximum weight matching are cumbersome to implement and impractical for large switches. This motivates us to explore service trace control policies which yield MSL-like performance at manageable complexity.

Remark 3: Step 2 of MSL can be generalized to construct a broader class of policies, namely MSL(ℓ), indexed

by $\ell \in \mathbb{N} \cup \{0\}$. Under MSL($\ell$)[1], a VOQ served by the chosen complete configuration is idled only when its deviation is ℓ or more. By this token, MSL $\equiv$ MSL(0).

III. Subset Based Service Trace Control

To address the issue of high computational complexity associated with optimal service trace control, we propose a *subset based control* approach in this section. The key idea is to partition the configuration set $\mathcal{V}$ of size $N!$ into smaller disjoint subsets of size N each and operate the switch using configurations from only one of these subsets in any time-slot.

A. Subset construction

It follows by design that all configuration vectors for an IQ switch are of the form $\mathbf{v} = [\mathbf{e}_{\pi(1)}\ \mathbf{e}_{\pi(2)} \dots \mathbf{e}_{\pi(N)}]$, where π is a permutation of $\{1, 2, \dots, N\}$ and $\mathbf{e}_i$ is as defined in Section I-D. Now, we define the *circular shift operator*.

Definition 3: The **circular shift operator** $\mathcal{C} : \mathcal{V} \mapsto \mathcal{V}$ is given by

$$\mathcal{C}(\mathbf{v}) \triangleq [\mathbf{e}_{\pi(N)}\ \mathbf{e}_{\pi(1)}\ \mathbf{e}_{\pi(2)} \dots \mathbf{e}_{\pi(N-1)}], \tag{15}$$

where $\mathbf{v} = [\mathbf{e}_{\pi(1)}\ \mathbf{e}_{\pi(2)} \dots \mathbf{e}_{\pi(N)}] \in \mathcal{V}$ is a switch configuration vector.

Recursively define $\mathcal{C}^k(\mathbf{v}) \triangleq \mathcal{C}(\mathcal{C}^{k-1}(\mathbf{v}))$, $k \in \mathbb{N}$, which corresponds to applying the circular shift operator k times to $\mathbf{v}$. By convention, $\mathcal{C}^0(\mathbf{v}) = \mathbf{v}$. Also, note that $\mathcal{C}^k(\mathbf{v}) = \mathcal{C}^{k \mod N}(\mathbf{v})$. Thus, starting with any configuration vector $\mathbf{v} \in \mathcal{V}$, we can generate a set of N distinct configuration vectors by applying the operator $\mathcal{C}$ to $\mathbf{v}$ $N-1$ times. We say that $\mathbf{v}$ *generates* the *configuration subset*

$$\mathcal{S}_{\mathbf{v}} = \{\mathbf{v}, \mathcal{C}(\mathbf{v}), \dots, \mathcal{C}^{N-1}(\mathbf{v})\} \subset \mathcal{V} \tag{16}$$

and refer to $\mathbf{v}$ as the *generator vector*. Following the outlined procedure, we can partition $\mathcal{V}$ into $(N-1)!$ disjoint configuration subsets of size N each. As an example, the configuration subsets for a 3×3 switch are depicted in Fig. 3.

For any $\mathbf{v} \in \mathcal{V}$, $\langle \mathbf{v}, \mathcal{C}^k(\mathbf{v}) \rangle = 0\ \forall\ k \in \mathbb{N}$. Physically, this implies that no VOQ is served by more than one configuration in a subset. Geometrically, this mean that all configuration vectors within a subset are "orthogonal" to each other. We therefore say that the generated subsets are *orthogonal*. Also, note that every VOQ is served by some configuration within a subset. Consequently, we say that every subset is *complete*. Combining the orthogonality and completeness properties we see that every VOQ is associated with *exactly* one configuration vector in every subset, implying

$$\sum_{j=0}^{N-1} \mathcal{C}^j(\mathbf{v}) = \mathbf{1} \quad \forall\ \mathbf{v} \in \mathcal{V}. \tag{17}$$

B. The MSL-SS policy

Let $\mathcal{S}_{\mathbf{v}} = \{\mathcal{C}^i(\mathbf{v})\}_{i=0}^{N-1}$ be the configuration subset generated by $\mathbf{v}$. Now consider operating the switch such that the service trace controller is allowed to choose configurations from $\mathcal{S}_{\mathbf{v}}$ alone, rather than from $\mathcal{V}$. In particular, consider a restriction of the MSL policy of Section II-E to the configuration subset $\mathcal{S}_{\mathbf{v}}$. We get the following two-step policy, which we call the **Maximum Sum of Lags - Single Subset** (MSL-SS) policy:

1) Select configuration $\mathcal{C}^{i^\star}(\mathbf{v}) \in \mathcal{S}_{\mathbf{v}}$ such that

$$i^\star = \underset{i=0,\dots,N-1}{\arg\min} \left\{ \langle \mathbf{d} - \mathbf{x}^t, \mathcal{C}^i(\mathbf{v}) \rangle \right\}. \tag{18}$$

2) Extract a partial configuration from $\mathcal{C}^{i^\star}(\mathbf{v})$ by idling all VOQs with non-negative deviation.

[1]Note that it may not be feasible to realize MSL(ℓ) for arbitrary $\ell > 0$, if the switch cannot provide a lead of ℓ even in the absence of congestion, due to unavailability of packets ahead of their departure times. However, MSL(ℓ) is pertinent in a scenario where the switch resides at the egress of a multimedia server, where all traffic streams are pre-cached at the input of the switch. In this case, the switch can furnish a lead of up to ℓ to provide a "cushion" against possible congestion in the downstream network.

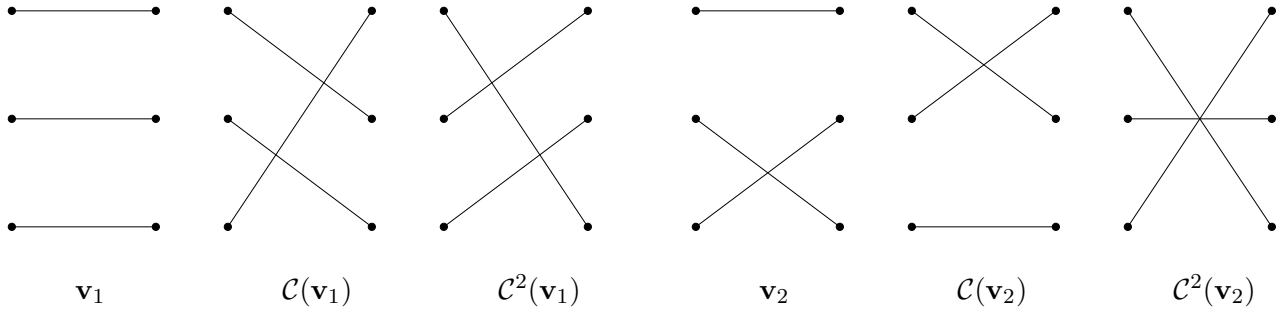

Fig. 3. Two configuration subsets (of size 3 each) for a 3×3 switch. The three leftmost configurations are generated by $\mathbf{v}_1 = [\mathbf{e}_1\ \mathbf{e}_2\ \mathbf{e}_3]$ and the three rightmost configurations are generated by $\mathbf{v}_2 = [\mathbf{e}_1\ \mathbf{e}_3\ \mathbf{e}_2]$.

The per time-slot computational complexity for MSL-SS is $\mathcal{O}(N^2)$, in contrast to $\mathcal{O}(N^3)$ for MSL.

Remark 4: To compute the optimal decision for MSL-SS, N inner products of the form $\langle \mathbf{d} - \mathbf{x}^t, \mathcal{C}^i(\mathbf{v}) \rangle$ need to be computed, followed by a $\min$ of the resulting N numbers. Each of these inner products involve vectors of length N^2. However, note that one of the vectors involved in each inner product is a configuration vector, which is relatively sparse (only N out of the N^2 entries are non-zero). Further, all the non-zero entries are equal to 1. Each inner product, $\langle \mathbf{d} - \mathbf{x}^t, \mathcal{C}^i(\mathbf{v}) \rangle$, is therefore simply a sum of N numbers. The MSL-SS policy is thus straightforward to implement, compared to algorithms used for computing maximum weight matching (needed for MSL).
Two crucial questions arise at this point:

1) What is the performance loss (if any) incurred by operating the switch using only one configuration subset?
2) Can we compensate for the loss (if needed), without sacrificing the advantage of low complexity?

We will address these questions in the remainder of the paper.

IV. Meta-Queue Based Service Trace Control

In this section, we study subset based service trace control in a broader framework, based on the notion of meta-queues. We will recover the MSL-SS policy proposed in Section III-B as a special case of the meta-queue framework.

A. Meta-queue construction

Setting the switch in the complete configuration given by $\mathbf{v} = [\mathbf{e}_{\pi(1)} \dots \mathbf{e}_{\pi(N)}]$ is equivalent to serving VOQs indexed by the set $\mathcal{I} = \{(i-1)N + \pi(i),\ i = 1, \dots, N\}$. Thus, every complete configuration serves N VOQs concurrently, which we "group" together to form a *meta-queue*.

Let us focus on a single subset, say $\mathcal{S}_{\mathbf{v}}$. Since $\mathcal{S}_{\mathbf{v}}$ is orthogonal and complete by construction, each configuration in $\mathcal{S}_{\mathbf{v}}$ can be associated with a unique meta-queue, constructed by "grouping" N distinct VOQs. Note that all N^2 VOQs are assigned to some meta-queue, each one exactly once. The head of line (HOL) *meta-packet* of a meta-queue is constructed by grouping the HOL packets of its N constituent VOQs. With this construction, choosing a switch configuration is equivalent to serving the HOL meta-packet of the corresponding meta-queue.

While grouping concurrently served VOQs to form a meta-queue seems quite natural, the relation between the deviation of a meta-queue and the deviations of its constituent VOQs is not immediately evident. In fact, we have the freedom to choose a mapping $\Gamma : \mathbb{Z}^N \mapsto \mathbb{Z}$, which relates the deviation of a meta-queue to the deviations of its N constituent VOQs. Given a mapping Γ, the problem of subset based control of an IQ switch turns into a problem of scheduling N parallel meta-queues on a single server. The latter is an important and interesting scheduling problem in its own right (e.g. see [23]).

We now briefly digress from the service trace control problem for the IQ switch to study the single server scheduling problem mentioned above. Subsequently, we will show that by appropriately choosing Γ, one can construct good, low complexity service trace control policies for an IQ switch.

B. The single server scheduling problem

The formulation is similar in spirit to the formulation for an IQ switch (Section II-B), and so is the notation. Consider a system comprised of $N+1$ parallel meta-queues and a single server. The i^{th} meta-queue is denoted $\mathcal{M}_i$, $i = 0, \ldots, N$. In every time-slot the scheduler serves the HOL meta-packet of one of the meta-queues, chosen according to some scheduling policy. While $\mathcal{M}_1, \ldots, \mathcal{M}_N$ are "physical" meta-queues, $\mathcal{M}_0$ is a "dummy" meta-queue, scheduling which is tantamount to idling the server. Each meta-queue is associated with a traffic stream characterized by a target service profile (TSP). The interpretation of the TSP in this context is identical to Section II-A, i.e., it specifies the time-slots in which meta-packets from a meta-queue should ideally depart the server. The TSP associated with $\mathcal{M}_0$ has all zero entries. We denote by $\widetilde{\mathbf{d}}^t = (\widetilde{d}_1^t, \ldots, \widetilde{d}_N^t)$ the deviation vector for the system in the t^{th} time-slot, where $\widetilde{d}_i^t$ is the deviation for $\mathcal{M}_i$. Define $\widetilde{\mathbf{x}}^t \triangleq (\widetilde{s}_1^t, \ldots, \widetilde{s}_N^t)$ and $\widetilde{\mathbf{X}}^t \triangleq (\widetilde{S}_1^t, \ldots, \widetilde{S}_N^t)$, where $\widetilde{s}_i^t$ and $\widetilde{S}_i^t$ are respectively the t^{th} elements of the TSP and cumulative TSP (cTSP) of $\mathcal{M}_i$. To $\mathcal{M}_i$ we assign the cost function $\psi_i(k)$, which quantifies the cost of deviation $k \in \mathbb{Z}$. Similar to Section II-B, we assume that $\psi_i(k)$ is non-negative, convex, and increasing for both $k > 0$ and $k < 0$, and $\psi_0(k) = 0 \; \forall \; k$. Finally, let

$$\Psi(\mathbf{d}^t) \triangleq \sum_{i=1}^{N} \psi_i(d_i^t) \tag{19}$$

denote the sum of deviation costs of all meta-queues.

We confine our attention to a finite horizon of T time-slots. At the beginning of every time-slot, the scheduler selects one of the $N+1$ meta-queues for service. The configuration vector corresponding to scheduling $\mathcal{M}_i$ is $\mathbf{e}_i$. An admissible policy $\widetilde{\Pi}_T$ for the single server scheduling problem is a sequence of scheduling decisions $\{i_t\}_{t=1}^T$, corresponding to scheduling meta-queue $\mathcal{M}_{i_t}$ in the t^{th} time-slot. Let $\widetilde{\mathbf{d}}_{\widetilde{\Pi}_T}^t$ denote the deviation vector at the beginning of the t^{th} time-slot under scheduling policy $\widetilde{\Pi}_T$. Our goal is to compute the optimal finite horizon policy $\widetilde{\Pi}_T^\star$ which satisfies

$$\widetilde{\Pi}_T^\star = \arg\min_{\widetilde{\Pi}_T} \left\{ \sum_{t=1}^{T} \Psi(\widetilde{\mathbf{d}}_{\widetilde{\Pi}_T}^t) \right\}. \tag{20}$$

We specify the state of the system at the end of the t^{th} time-slot by

$$\mathbf{n}^t = (n_1^t, \ldots, n_N^t), \tag{21}$$

where n_i^t is the number of times $\mathcal{M}_i$ has been served within the first t time-slots. Since the server is allowed to idle, $\langle \mathbf{n}^t, \mathbf{1} \rangle \leq t \; \forall \; t$. The system state and deviation vector are uniquely related by

$$\widetilde{\mathbf{d}}^{t+1} = \widetilde{\mathbf{d}}^0 + \mathbf{n}^t - \widetilde{\mathbf{X}}^t. \tag{22}$$

If the state at the beginning of the t^{th} time-slot is $\mathbf{n}$ and the scheduler chooses $\mathcal{M}_i$ in the t^{th} time-slot, the new state at the beginning of the $(t+1)^{st}$ time-slot is $\mathbf{n} + \mathbf{e}_i$. Letting $\widetilde{V}^t(\mathbf{n})$ denote the cost-to-go at the beginning of the t^{th} time-slot in state $\mathbf{n}$, we have the following DP equations for $t = 1, \ldots, T$

$$\widetilde{V}^t(\mathbf{n}) = \min_{i=0,\ldots,N} \left\{ \widetilde{V}^{t+1}(\underbrace{\mathbf{n} + \mathbf{e}_i}_{\text{New state}}) + \Psi(\underbrace{\mathbf{n} + \mathbf{e}_i - \widetilde{\mathbf{X}}^t}_{\text{New deviation vector}}) \right\}, \tag{23}$$

and the boundary conditions $\widetilde{V}^{T+1}(\mathbf{n}) = 0 \; \forall \; \mathbf{n}$. For notational convenience, define

$$\Omega^t(\mathbf{n}) \triangleq \widetilde{V}^t(\mathbf{n}) + \Psi(\mathbf{n} - \mathbf{X}^t). \tag{24}$$

Also, define the **pairwise decision functions**

$$\gamma_{ij}^t(\mathbf{n}) \triangleq \Omega^{t+1}(\mathbf{n} + \mathbf{e}_i) - \Omega^{t+1}(\mathbf{n} + \mathbf{e}_j), \quad i \neq j. \tag{25}$$

It follows that $\widetilde{\Pi}_T^\star$ "prefers" $\mathcal{M}_i$ over $\mathcal{M}_j$ in the t^{th} time-slot in state $\mathbf{n}$ if $\gamma_{ij}^t(\mathbf{n}) \leq 0$, and "prefers" $\mathcal{M}_j$ else. The pairwise decision functions satisfy the following:

Lemma 1 (Monotonicity of γ): $\gamma_{ij}^t(\mathbf{n})$ is a non-decreasing function of n_i and a non-increasing function of n_j for $i, j \in \{0, \ldots, N\}$, $i \neq j$, and $t = 1, \ldots, T$.

Proof: See Appendix VIII-B. ■

Lemma 1 can be used to show that any two-dimensional subspace of the N-dimensional state-space is partitioned into $N+1$ connected *decision regions* by $\widetilde{\Pi}_T^\star$. The states in the i^{th} decision region are those in which $\widetilde{\Pi}_T^\star$ schedules $\mathcal{M}_i$ in the t^{th} time-slot. Further, for every t, as n_i^t increases for fixed n_j^t, $j \neq i$, $\widetilde{\Pi}_T^\star$ *switches over* from $\mathcal{M}_i$ to $\mathcal{M}_k$ for some $k \neq i$. Thereafter, $\widetilde{\Pi}_T^\star$ *never* switches back to $\mathcal{M}_i$. Unfortunately, this neat structural insight does not immediately yield a low complexity approximation of the optimal policy.

Example 9: Consider a system with three meta-queues ($N = 3$) and a time-horizon of $T = 40$ time-slots. Let us fix $n_3^t = 8$ and look at the projection of the three dimensional state-space on the (n_1^t, n_2^t) plane, for $t = 30$. The entries in the TSPs of the meta-queues were generated from an i.i.d. Bernoulli process with parameter p. Fig. 4 and Fig. 5 depict the partitioning of the (n_1, n_2) plane for $p = 0.1$ and $p = 0.3$, respectively. Since the TSPs are more sparse in the case $p = 0.1$ (more relaxed deadlines), it is optimal to idle the server in several states.

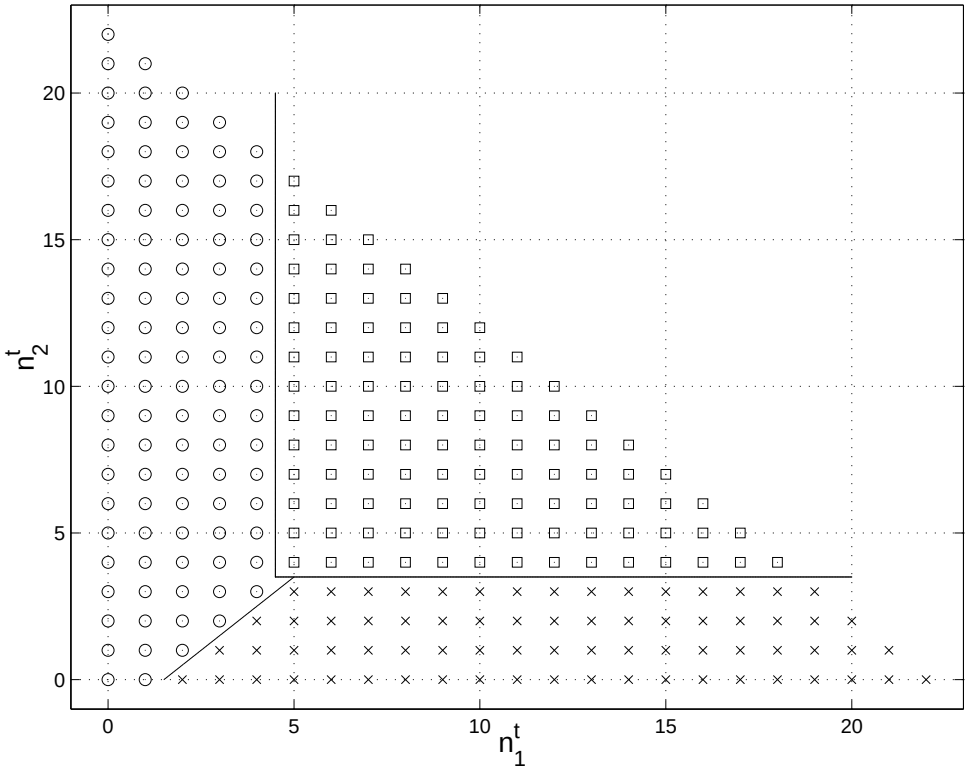

Fig. 4. Partitioning of the (n_1^t, n_2^t) plane into decision regions for fixed $n_3^t = 8, T = 40, t = 30$ for $p = 0.1$. The states in which it is optimal to schedule $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2$, and $\mathcal{M}_3$ are depicted by $\square$, $\circ$, $\times$, and $\star$ respectively.

C. The myopic/greedy policy

The complexity of computing the optimal policy $\widetilde{\Pi}_T^\star$ increases exponentially in both T and N. However, the per time-slot complexity of the myopic/greedy policy associated with the problem is only $\mathcal{O}(N)$. The myopic policy schedules $\mathcal{M}_{i^\star}$ in the t^{th} time-slot in state $\mathbf{n}$ such that

$$i^\star = \underset{j=0,\ldots,N}{\arg\min}\{\Psi(\mathbf{n} + \mathbf{e}_j - \widetilde{\mathbf{X}}^t)\}. \tag{26}$$

Once again, the myopic policy for the aforementioned scheduling problem is provably optimal for the case $N = 2$. The proof is similar to the proof of Theorem 1 and is omitted.

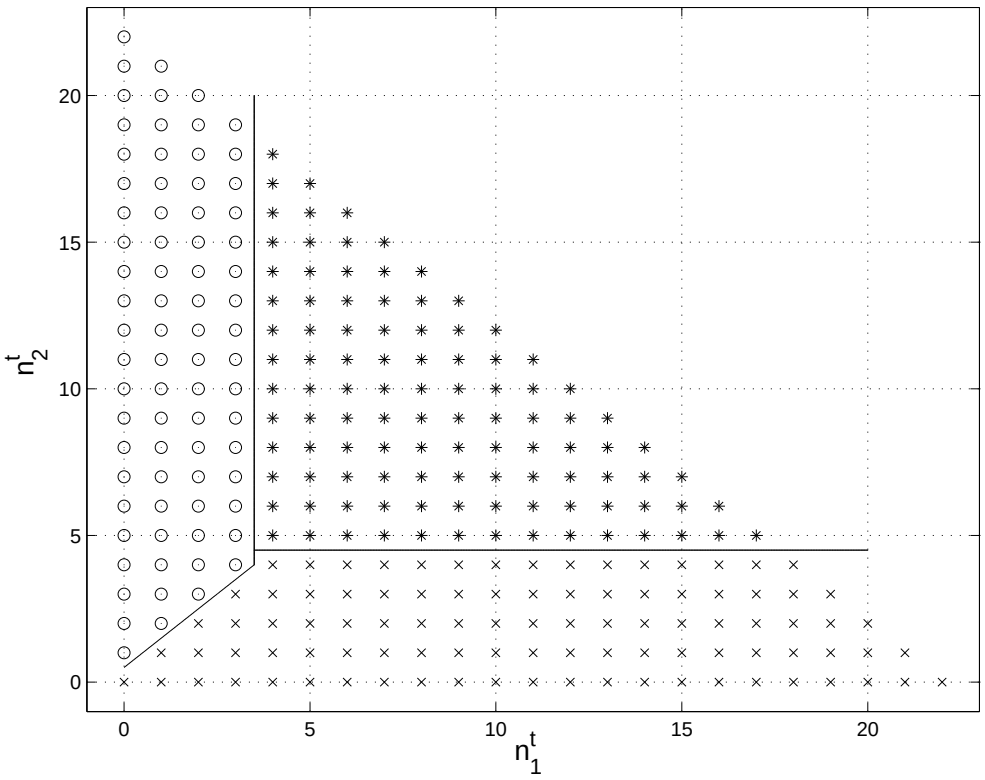

Fig. 5. Partitioning of the (n_1^t, n_2^t) plane into decision regions for fixed $n_3^t = 8, T = 40, t = 30$ for $p = 0.3$. The states in which it is optimal to schedule $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2$, and $\mathcal{M}_3$ are depicted by $\square$, $\circ$, $\times$, and $\star$ respectively.

D. The Largest Lag First (LLF) policy

Consider the special case of quadratic cost functions for concreteness. It is easily seen that the myopic policy in this case selects $\mathcal{M}_{i^\star}$ such that

$$i^\star = \begin{cases} \tilde{i}^\star & ; \quad 2(\widetilde{d}_{\tilde{i}^\star} - \widetilde{x}^t_{\tilde{i}^\star}) + 1 < 0 \\ 0 & ; \quad \text{else,} \end{cases} \tag{27}$$

where

$$\tilde{i}^\star \triangleq \underset{j=1,\ldots,N}{\arg\min} \{\widetilde{d}_j - \widetilde{x}^t_j\}. \tag{28}$$

The arguments are similar to those given in Example 6.

We refer to the policy in (27) as **Largest Lag First** (LLF), because it chooses the meta-queue with the most negative deviation (equivalently, largest lag). The LLF policy idles the server if the deviations of all meta-queues are non-negative.

E. Meta-queue based service trace control

Having studied the salient features of the single server scheduling problem, we revert our attention to the service trace control problem for the IQ switch. Suppose that the STC chooses a configuration from subset $\mathcal{S}_{\mathbf{v}}$ alone, and VOQ deviations are mapped to meta-queue deviations through a mapping Γ. Let $\mathcal{I}_i = \{i_1, \ldots, i_N\}$ denote the set of VOQs which are served by the i^{th} configuration in $\mathcal{S}_{\mathbf{v}}$, namely $\mathcal{C}^{i-1}(\mathbf{v})$. These VOQs constitute the i^{th} meta-queue in the single server system. Given deviation vector $\mathbf{d}$ for the switch, we denote the deviation of the i^{th} meta-queue by $\Gamma(\mathbf{d}; \mathcal{I}_i)$. The LLF policy of Section IV-D then schedules the j^{th} meta-queue such that

$$j = \underset{i=1,\ldots,N}{\arg\min} \{\Gamma(\mathbf{d}; \mathcal{I}_i)\}. \tag{29}$$

Once a meta-queue is chosen, a partial configuration is extracted by idling VOQs with non-negative deviation. We now examine two special choices of Γ.

1) The MSL-SS policy revisited: Consider $\Gamma(\mathbf{d};\mathcal{I}) = \sum_{j\in\mathcal{I}} d_j$; With this choice of Γ, we recover the MSL-SS policy proposed in Section III-B. Thus, the service trace control problem for an IQ switch with single subset operation is equivalent to the single server scheduling problem if all cost functions (both for the VOQs and the meta-queues) are quadratic and the deviation of a meta-queue is defined as the sum of the deviations of its constituent VOQs.

2) The LLF-SS policy: Consider $\Gamma(\mathbf{d};\mathcal{I}) = \min_{j\in\mathcal{I}}\{d_j\}$; For this Γ, the STC in effect selects the VOQ with the largest lag. Since every VOQ is associated with a unique configuration in $\mathcal{S}_{\mathbf{v}}$, selecting a VOQ immediately identifies a unique switch configuration. We call this policy the **Largest Lag First - Single Subset** (LLF-SS) policy. The per time-slot complexity of LLF-SS is $\mathcal{O}(N^2)$, since it involves computing the maximum of an unsorted list of N^2 numbers. It can be reduced to $\mathcal{O}(N)$ through parallelization or efficient data structures (for maintaining dynamic lists).

Remark 5: There is a natural interpretation for the above choice of Γ. Suppose that the switch is operated using only complete configurations and has been set in configuration $\mathcal{C}^{i-1}(\mathbf{v})$ τ times by the end of the t^{th} time-slot. Denote by $S^t_{i_j}$ the t^{th} entry of the TSP of $\mathcal{Q}_{i_j}$. It follows that the deviation of the $\mathcal{Q}_{i_j}$ in the t^{th} time-slot is $d^t_{i_j} = \tau - S^t_{i_j}$, since $\mathcal{C}^{i-1}(\mathbf{v})$ serves VOQs indexed by $\mathcal{I}_i = \{i_1,\ldots,i_N\}$. Now, define

$$\widetilde{S}^t_i \triangleq \max_{j=1,\ldots,N}\{S^t_{i_j}\}. \tag{30}$$

If configuration $\mathcal{C}^{i-1}(\mathbf{v})$ is chosen at least $\widetilde{S}^t_i$ times by the end of the t^{th} time-slot ($\tau \geq S^t$), all N VOQs indexed by set $\mathcal{I}_i$ have a non-negative deviation (no lag). We therefore let $\widetilde{\mathbf{S}}_i = (\widetilde{S}^1_i, \widetilde{S}^2_i, \ldots)$ be the cTSP of the meta-queue generated by grouping VOQs indexed by set $\mathcal{I}_i$. It follows that the deviation of the i^{th} meta-queue in the t^{th} time-slot is $\widetilde{d}^t_i = \tau - \max_{j=1,\ldots,N}\{S^t_{i_j}\}$, i.e., $\widetilde{d}^t_i = \min_{j=1,\ldots,N}\{d^t_{i_j}\}$. In words, the deviation of a meta-queue is the *minimum* of the deviation of its constituent VOQs.

Remark 6: The meta-queue construction provides a general framework for designing service trace control policies under single subset operation. While we have illustrated the idea with two specific examples here, different families of policies with varying performance tradeoffs can be constructed by appropriately selecting the mapping Γ, as well as the meta-queue selection policy. For instance, we can set $\Gamma(\mathbf{d};\mathcal{I}) = \sum_{j\in\mathcal{I}} c_j d_j$, where $\{c_j\} \geq 0$ are weight parameters chosen to provide differentiated QoS to VOQs.

V. Admissible Region and Subset Selection

Consider a traffic stream with target stream profile (TSP) $\mathbf{s}$ and the corresponding cumulative TSP $\mathbf{S}$. The average "distance" between consecutive "1"s in $\mathbf{s}$ can be interpreted as the *average packet inter-departure time target* associated with this traffic stream. By definition, S^t is the number of "1"s in the TSP in the first t time-slots. We assume that the limit

$$\lambda = \lim_{t\to\infty} \frac{S^t}{t} \tag{31}$$

exists for every traffic stream, and refer to $1/\lambda$ as the average packet inter-departure time (IDT) target for the traffic stream. Going back to Example 3, where we considered periodic traffic with period δ, we see that $S^t = \lfloor t/\delta \rfloor$ and $\lambda = 1/\delta$.

A larger λ implies smaller IDT targets on an average, which means the stream requires more service from the switch. Thus, λ can be thought of as the **load** imposed by a stream on the switch. Letting λ_i denote the load imposed by the stream at the i^{th} VOQ, we define

$$\boldsymbol{\lambda} \triangleq (\lambda_1,\ldots,\lambda_{N^2}) \tag{32}$$

as the **load vector** for the switch. We now consider a special case where the IDT targets for the traffic stream associated with the i^{th} VOQ are *geometrically distributed* [2] with parameter $\lambda_i \in (0,1)$. Equivalently, every entry in the TSP of $\mathcal{Q}_i$ is an independent identically distributed (i.i.d.) Bernoulli random variable [3] with mean λ_i. We

[2]For a geometrically distributed random variable X with parameter p, the probability mass function is given by $\mathbb{P}[X = k] = (1-p)^{k-1}p,\ k \in \mathbb{N}$.

[3]For a Bernoulli random variable X with parameter p, the probability mass function is given by $\mathbb{P}[X = 0] = 1 - p$ and $\mathbb{P}[X = 1] = p$.

refer to this scenario as **i.i.d. loading**. Further, we refer to the scenario $\lambda_i = \lambda, \forall\, i$, i.e., $\boldsymbol{\lambda} = (\lambda/N^2)\mathbf{1}$ as **uniform i.i.d loading**[4]. Using the notation introduced in (6), the i.i.d. assumption implies:

$$\begin{aligned} \mathbb{E}[\mathbf{x}^t] &= \boldsymbol{\lambda}\ \forall\, t \\ \mathbb{E}[x_i^t x_j^t] &= \lambda_i \text{ if } j = i \text{ and } \mathbb{E}[x_i^t x_j^t] = 0 \text{ if } j \neq i,\ \forall\, t \\ \mathbb{E}[x_i^t x_j^\tau] &= 0\ \forall\, i, j \text{ if } \tau \neq t. \end{aligned} \tag{33}$$

Next, we define the **admissible region** as the set of all load vectors for which some service trace control policy guarantees finite lags to all VOQs, at all times. Alternatively, if the switch is subject to a load vector not contained in the admissible region, the lag of at least one VOQ grows without bound, regardless of the service trace control policy employed. The admissible region for an IQ switch is given by

$$\begin{aligned} \Lambda \triangleq \{\boldsymbol{\lambda} : \sum_{j=1}^{N} \lambda_{(i-1)N+j} < 1,\ \sum_{j=1}^{N} \lambda_{j(N-1)+i} < 1, \\ i = 1, \ldots, N,\ \lambda_i \in (0,1),\ i = 1, \ldots, N^2\}. \end{aligned} \tag{34}$$

A policy which ensures finite lags for all VOQs for all load vectors $\boldsymbol{\lambda} \in \Lambda$ is said to be 100% admissible. More formally,

Definition 4: A policy Π is 100% admissible if $\forall\ \boldsymbol{\lambda} \in \Lambda$, $\liminf_{t\to\infty} \mathbb{E}[d_i^t] > -\infty\ \forall\ i$, where the notation $\mathbb{E}^{\Pi}[\cdot]$ implies that the expectation is computed under policy Π. As a special case, a policy Π is 100% admissible under i.i.d. loading if it satisfies the aforementioned property for all i.i.d. load vectors in Λ.

Theorem 2: $\liminf_{t\to\infty} \mathbb{E}[d_i^t] > -\infty\ \forall\, i$ under the MSL(ℓ) policy, for *any admissible* i.i.d. load.

Proof: See Appendix VIII-C. ■

We are now ready to answer the two questions raised at the end of Section III regarding the efficacy of subset based control. Our answer to the first question is that by restricting operation to a single subset, not all load vectors in Λ can be supported. However, all uniform loads can be supported. In particular,

Theorem 3: $\liminf_{t\to\infty} \mathbb{E}[d_i^t] > -\infty\ \forall\, i$ under the LLF-SS policy, for any *admissible uniform* i.i.d load, independent of the choice of operational subset.

Proof: See Appendix VIII-D. ■

Theorem 4: $\liminf_{t\to\infty} \mathbb{E}[d_i^t] > -\infty\ \forall\ i$ under the MSL(ℓ)-SS policy, for any *admissible uniform* i.i.d. load, independent of the choice of operational subset.

Proof: See Appendix VIII-E. ■

It must be noted that there exists a non-empty subset of non-uniform load vectors in Λ (even near the "boundary" of Λ) under which LLF-SS guarantees bounded lags to all VOQs, if the operational subset is suitably chosen.

Example 10: Consider $\boldsymbol{\lambda} = (1-\epsilon, 0, 0, 0, 0, 1-\epsilon, 0, 1-\epsilon, 0)$ for a 3×3 switch, for some $\epsilon \in (0, 2/3)$. In this case, operating LLF-SS with the first subset (generated by $\mathbf{v}_1$) in Fig. 3 cannot guarantee bounded lags to all VOQs, while operating the same policy with the second subset (generated by $\mathbf{v}_2$) can guarantee bounded lags.

It is possible to construct non-uniform i.i.d. load vectors in Λ under which LLF-SS cannot guarantee bounded lags to all VOQs, irrespective of the choice of the operational subset.

Example 11: Consider $\boldsymbol{\lambda} = (c, 0, 0, 0, c/2, c/2, 0, c/2, c/2)$ where $c = 1 - \epsilon$ for some $\epsilon \in (0, 1/2)$. The LLF-SS policy cannot guarantee bounded lags to all VOQs, regardless of the choice of operational subset.
Similar examples can be constructed for the MSL(ℓ)-SS policy.

A. Randomized subset selection

As we saw in the previous section, service trace control based on single subset operation is not enough to support all admissible loads. However, subset based operation in conjunction with an appropriate subset selection policy can achieve the desired goal. We propose one such subset selection policy in this section. To this end, denote the

[4]The theory developed here can be extended to the case where TSP entries are generated from a Markov modulated Bernoulli process by considering multi-step drifts of the Lyapunov function (see, for example, [24].)

Policy	Brief description	Complexity	100% admissibile	Knows λ
MSL	Pick configuration (from $\mathcal{V}$) with maximum sum of VOQ lags	$\mathcal{O}(N^3)$	✓	×
MSL-SS	Pick configuration (single subset) with max sum of VOQ lags	$\mathcal{O}(N^2)$	×	×
LLF-SS	Pick configuration (single subset) with most lagged VOQ	$\mathcal{O}(N^2)$	×	×
MSL-RS	Randomized subset selection + MSL-SS	$\mathcal{O}(N^2)$	✓	✓
LLF-RS	Randomized subset selection + LLF-SS	$\mathcal{O}(N^2)$	✓	✓
MSL-pSEL(P)	Periodic subset selection (with period P) + MSL-SS	$\mathcal{O}(N^2)$	✓	×
LLF-pSEL(P)	Periodic subset selection (with period P) + LLF-SS	$\mathcal{O}(N^2)$	✓	×

TABLE I

KEY PROPERTIES OF SOME SERVICE TRACE CONTROL POLICIES PROPOSED IN THE PAPER.

k^{th} configuration vector by $\mathbf{v}_k$ and the corresponding generated subset by $\mathcal{S}_k = \{\mathcal{C}^i(\mathbf{v}_k)\}_{i=0}^{N-1}$. Consider the Birkoff von Neumann (BV) decomposition [13] of load vector $\boldsymbol{\lambda} \in \Lambda$ given by

$$\boldsymbol{\lambda} = \sum_{k=1}^{(N-1)!} \sum_{i=0}^{N-1} \zeta_{ik} \mathcal{C}^i(\mathbf{v}_k), \qquad \sum_{k=1}^{(N-1)!} \sum_{i=0}^{N-1} \zeta_{ik} = \zeta < 1. \tag{35}$$

Define a probability distribution on the subsets by

$$\theta_k \triangleq \frac{1}{\zeta} \sum_{i=0}^{N-1} \zeta_{ik}, \; k = 1, 2, \ldots, (N-1)! \tag{36}$$

Now, consider the following two-step service trace control policy, namely **Maximum Sum of Lags - Random Subset** (MSL-RS), which combines the MSL-SS policy of Section III-B with the notion of randomized subset selection:

1) Select configuration subset $\mathcal{S}_k$ with probability θ_k.
2) Select a configuration from $\mathcal{S}_k$ based on MSL-SS.

The computational complexity of MSL-RS is $\mathcal{O}(N^2)$ per time-slot, since MSL-SS has complexity $\mathcal{O}(N^2)$ and the BV decomposition of $\boldsymbol{\lambda}$ contains at most $N^2 - 2N + 2$ non-zero terms [13].

Theorem 5: $\liminf_{t\to\infty} \mathbb{E}[d_i^t] > -\infty \; \forall \; i$ under MSL(ℓ)-RS, for *any admissible* i.i.d. load.

Proof: See Appendix VIII-F. ■

The **Largest Lag First - Random Subset** (LLF-RS) policy is constructed analogously, by combining the idea of randomized subset selection with the LLF-SS policy of Section IV-E.2. Finally, note that the MSL-RS and LLF-RS policies can be extended to construct the MSL(ℓ)-RS and LLF(ℓ)-RS families of policies, respectively (discussed in Remark 3). These policies allow traffic streams to enjoy a lead of up to $\ell > 0$, instead of idling them when they are not lagging.

B. Periodic subset selection

1) The pSEL(P) rule: The MSL(ℓ)-RS policy proposed in the previous section can support all admissible loads *and* has low computational complexity. However, the policy requires a priori knowledge of the load vector $\boldsymbol{\lambda}$. We, on the other hand, are interested in designing robust control policies which do not rely on statistical knowledge about the input traffic streams. Thus, to eliminate dependence on $\boldsymbol{\lambda}$, we propose the following *periodic* subset selection rule: Suppose the switch is currently being operated using configuration subset $\mathcal{S}_{\mathbf{v}}$. Every $P > 0$ time-slots, a complete configuration $\mathbf{v}^\star$ is selected, based on some service trace control policy. If $\mathbf{v}^\star \in \mathcal{S}_{\mathbf{v}}$, the switch continues to operate with configuration subset $\mathcal{S}_{\mathbf{v}}$, otherwise the switch starts operating in the configuration subset generated by $\mathbf{v}^\star$, viz., $\mathcal{S}_{\mathbf{v}^\star} = \{\mathbf{v}^\star, \mathcal{C}(\mathbf{v}^\star), \ldots, \mathcal{C}^{N-1}(\mathbf{v}^\star)\}$. Once a configuration subset has been selected, the switch can be operated using any subset based service trace control policy (e.g. MSL-SS). We refer to this subset selection rule as **pSEL(P)** (**P**eriodic **Sel**ection with period P).

2) The MSL-pSEL(P) policy: We combine the pSEL(P) subset selection rule with the MSL policy of Section II-E and the MSL-SS policy of Section III-B to propose the **Maximum Sum of Lags - Periodic Selection (P)** (MSL-pSEL(P)) service trace control policy. Every P time-slots, the MSL-pSEL(P) policy computes the switch configuration $\mathbf{v}^\star$ based on the MSL policy. If $\mathbf{v}^\star$ is in the current operational subset, the MSEL-pSEL(P) policy continues to operate the switch using the current subset, otherwise it switches to the configuration subset generated by $\mathbf{v}^\star$, viz., $\{\mathbf{v}^\star, \mathcal{C}(\mathbf{v}^\star), \ldots, \mathcal{C}^{N-1}(\mathbf{v}^\star)\}$. In the intermediate $P-1$ time-slots, MSL-pSEL(P) operates the switch using the MSL-SS policy.

The per time-slot complexity of the MSL policy is $\mathcal{O}(N^3)$. This computation needs to be done every P time-slots to update the configuration subset. The complexity of the MSL-SS policy is $\mathcal{O}(N^2)$, as discussed in III-B. The MSL-SS policy needs to be executed in the $P-1$ intermediate time-slots between configuration subset updates. Thus, the computational complexity of MSL-pSEL(P) is $\mathcal{O}\left(N^3/P + (1-1/P)N^2\right)$. If $P = \mathcal{O}(N)$, i.e., if the configuration subset is updated roughly every N time-slots for a switch of size $N \times N$, the complexity of MSL-pSEL(P) is $\mathcal{O}(N^2)$.

MSL-pSEL(P) has all the desired traits - a manageable complexity of $\mathcal{O}(N^2)$, no dependence on load vector $\boldsymbol{\lambda}$, and as Theorem 6 tells us, it is 100% admissible under i.i.d. loading.

Theorem 6: $\liminf_{t\to\infty} \mathbb{E}[d_i^t] > -\infty \; \forall \; i$ under MSL-pSEL(P), for *any admissible* i.i.d. load.

Proof: See Appendix VIII-G. ■

We close this section by noting that the **Largest Lag First - Periodic Selection (P)** (LLF-pSEL(P)) policy is constructed by combining the LLF policy (Section IV-D) and the LLF-SS policy (Section IV-E.2) with the pSEL(P) rule.

VI. Performance Evaluation

In this section, we present simulation results to characterize the performance of the proposed service trace control policies.

A. Simulation setup

All results presented here are for a 16×16 IQ switch. We contrast the performance of the following policies:

- *Maximum Sum of Lags (MSL)*: This policy was proposed in Section II-E. MSL computes the maximum weight matching (with VOQ lags as edge weights) over all possible switch configurations (set of size $N!$), and will be the benchmark for all other lower complexity policies.
- *Maximum Sum of Lags - Single Subset (MSL-SS)*: This policy was proposed in Section III-B. MSL-SS computes the maximum weight matching over one configuration subset only (using VOQ lags as edge weights).
- *Largest Lag First - Single Subset (LLF-SS)*: This policy was proposed in Section IV-E.2. LLF-SS operates on a single configuration subset, and picks the VOQ (and hence the configuration, since each VOQ is associated with a unique configuration within a subset) with the largest lag.
- *Maximum Sum of Lags - Periodic Selection (MSL-pSEL(16))*: This policy was proposed in Section V-B.2. The behavior is similar to MSL-SS, except that the underlying operational configuration subset is updated every 16 slots.
- *Largest Lag First - Periodic Selection (MSL-pSEL(16))*: This policy was proposed in Section V-B.2. The behavior is similar to LLF-SS, except that the underlying operational configuration subset is updated every 16 slots.

Salient features of the above policies are enumerated in Table I. All policies require a two-step implementation. A complete switch configuration is selected in the first step and all VOQs with non-negative deviations are idled to extract a partial configuration in the second step. Thus, no VOQ can lead under any of the policies under consideration. We also simulated the performance of policies which allow VOQs to acquire a lead of up to $\ell > 0$, but did not observe any relative difference in the performance of different policies for fixed ℓ.

We consider the following two performance metrics:

- *Average Deviation*: Empirical mean of VOQ deviations, averaged over all $16^2 = 256$ VOQs.
- *Variance*: Empirical variance of VOQ deviations, averaged over all 256 VOQs.

Since no VOQ can lead in our simulation setup, the deviation of every VOQ (and hence the average deviation for every policy) is upper bounded by zero.

Clearly, we want both the mean and variance of the deviations to be close to zero, for all traffic streams. A mean close to zero with large variance is not sufficient, since it indicates severe instantaneous positive/negative distortion of the target profiles. In other words, it implies that the output of the switch is not "smooth". This is not ideal from a flow control perspective, since a bursty output stream from the switch makes scheduling and buffering at downstream switches harder. Also, a small variance with a large non-zero mean is undesirable, since it indicates that one or more VOQs are missing their deadlines frequently.

Remark 7: Ideally, all policies should be benchmarked relative to the optimal service trace control policy, which is computed by solving the DP equations in (11). However, the complexity of evaluating the optimal policy grows exponentially with the size of the switch, viz. N and the length of the time horizon of interest, viz. T. In our setup, $N = 16$ and $T = 50,000$. The complexity of solving the DP equations for a problem of this magnitude is simply prohibitive. We therefore resort to the next best option, i.e., using the myopic policy (MSL) as a performance benchmark. Recall that we have analytically proven the optimality of the myopic policy for $N = 2$ (see Theorem 1) and numerically verified that it is "close" to being optimal for $N = 3, 4$. Note that even the myopic policy is quite expensive to implement, since it entails computing a maximum weight matching in every time-slot.

B. Discussion of simulation results

We report simulation results for four distinct loading scenarios. Every point on the performance curves depicted here was generated by averaging over 50,000 time-slots.

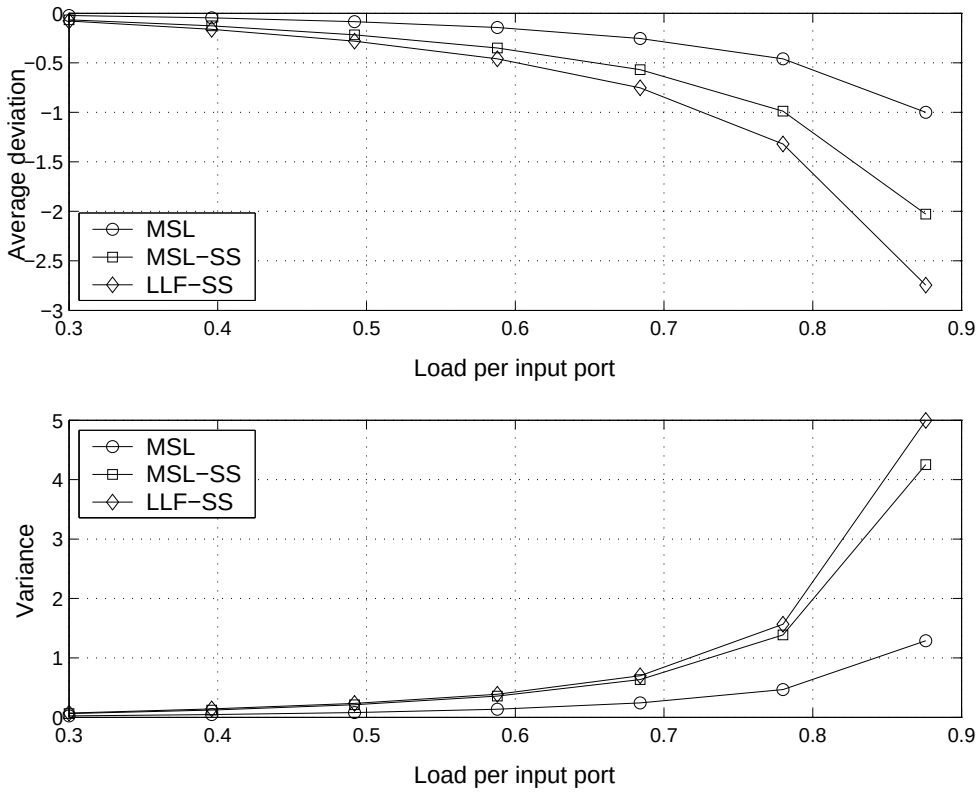

Fig. 6. Performance of MSL, MSL-SS and LLF-SS under uniform i.i.d. loading for an 16×16 switch

1) Uniform i.i.d. loading: In this case, all streams have geometrically distributed inter-departure time targets with parameter $\lambda/16$. The load per input port is therefore $16 \times \lambda/16 = \lambda$ (number of VOQs $\times$ load per stream/VOQ). The performance of the MSL, MSL-SS, and LLF-SS policies is depicted in Fig. 6. The results show that operating the switch with a single subset works quite well if the switch is uniformly loaded, especially up to $\sim 60\%$ loading of the switch. The loss in performance vis-a-vis the MSL policy comes with a significant reduction in complexity.

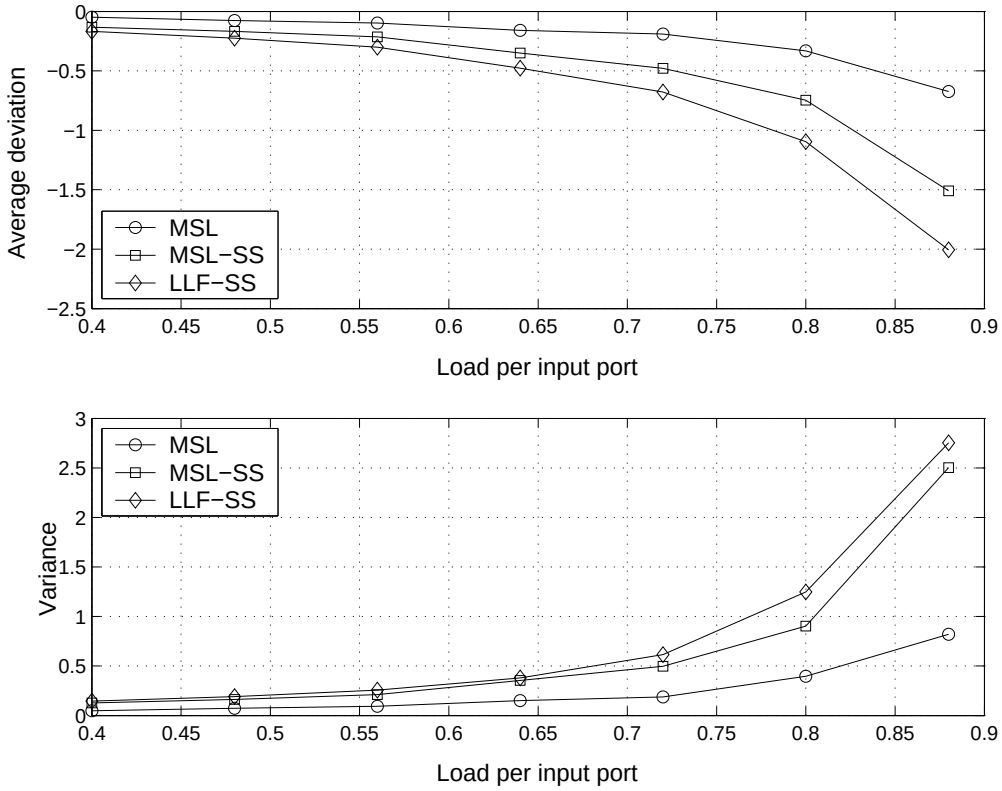

Fig. 7. Performance of MSL, MSL-SS and LLF-SS under parallel-heavy i.i.d. loading for an 16×16 switch

Recall that the MSL policy has to perform a full-blown maximum weight matching computation in every time-slot. Subset selection is not needed in the uniform loading scenario (Theorems 3 and 4). It suffices to operate the switch using only one configuration subset, which can be chosen arbitrarily.

2) Parallel-heavy i.i.d loading: In this case, all traffic streams associated with "parallel" VOQs have geometrically distributed inter-departure time targets with parameter λ_1 and "diagonal" VOQs have geometrically distributed inter-departure time targets with parameter $\lambda_2 < \lambda_1$. We call a VOQ parallel if it buffers packets destined from the i^{th} input port to the i^{th} output port for $i \in \{1, \ldots, 16\}$, and call it diagonal otherwise. Thus, a 16×16 switch has 16 parallel VOQs and $16^2 - 16 = 240$ diagonal VOQs. We fixed λ_2 and varied λ_1 to vary the load per input port, viz. $\lambda_1 + 15\lambda_2$. Performance results are depicted in Fig. 7. For MSL-SS and LLF-SS, we selected the subset generated by the configuration which concurrently serves all 16 parallel VOQs. Since this configuration needs to be selected frequently, especially as λ_1 increases, single subset operation based policies perform quite well in this non-uniform loading scenario. Note that subset based policies would perform poorly in this scenario if the configuration subset is not selected appropriately. Good performance can however be achieved by combining subset based operation with the periodic subset selection rule, as illustrated by the next set of simulation results.

3) Cross-heavy i.i.d. loading: Once again, all traffic streams have geometrically distributed inter-departure time targets. However, VOQs which buffer packets destined from input port i to output port $i + 1$ (for odd i) and from input port i to output port $i - 1$ (for even i) are more heavily loaded (parameter λ_1) than other VOQs (parameter $\lambda_2 < \lambda_1$). We allude to this scenario as cross-heavy loading because the configuration which serves the more heavily loaded VOQs forms a criss-cross pattern with eight "crosses" ($\times$). We fixed λ_2 and varied λ_1 to vary the load per input port, viz. $\lambda_1 + 15\lambda_2$. For MSL-SS and LLF-SS, we used the same configuration subset as for the parallel-heavy i.i.d. loading experiment. Since the cross pattern is not contained in this subset, the performance of single subset based policies degrades severely as λ_1 increases, and is therefore not depicted here. However, periodic subset selection, in conjunction with single subset operation delivers performance quite close to the benchmark MSL policy, especially for switch loading up to $\sim 65\%$. Recall that this performance is achieved at a much lower computational complexity. The results are depicted in Fig. 8. The MSL-SS and LLF-SS policies would have performed well if we had chosen the configuration subset which serves the most heavily loaded VOQs (the ones forming the cross

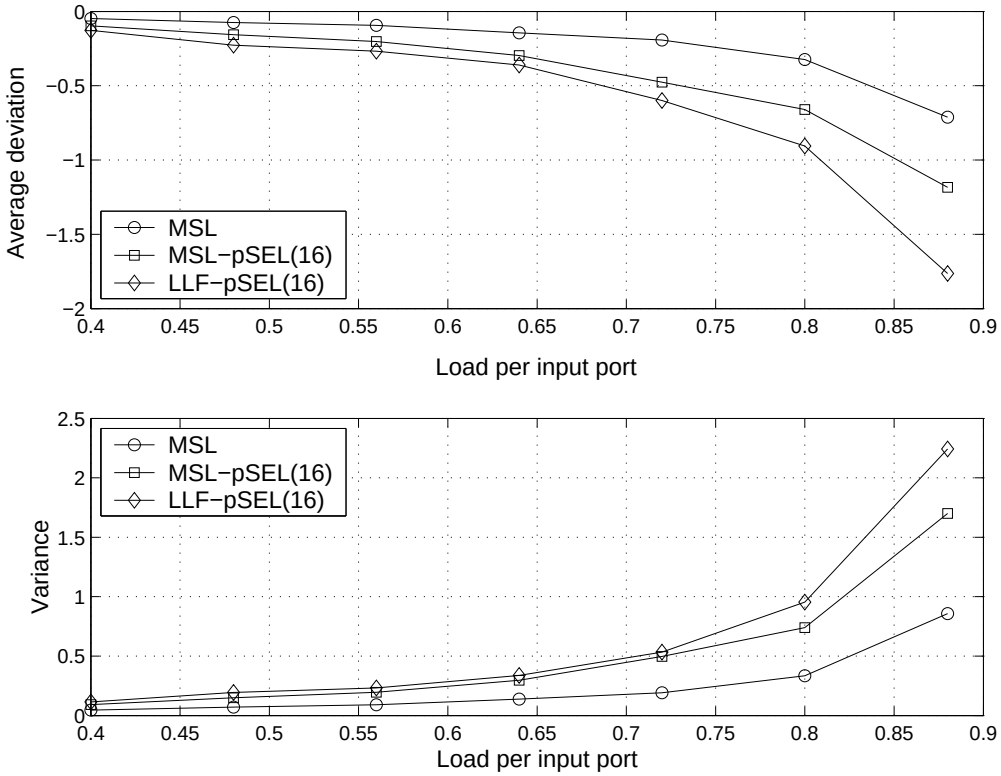

Fig. 8. Performance of MSL, MSL-pSEL(16) and LLF-pSEL(16) under cross-heavy i.i.d. loading for an 16×16 switch

pattern) as the operational subset. This is exactly what we had done in the diagonal loading scenario. However, it is not always possible for the switch controller to have a priori information about the loading pattern. Subset based operation therefore needs to be combined with a subset selection rule to ensure good performance in unknown loading scenarios.

4) Uniform periodic loading: In this case, all traffic streams have identical inter-departure time targets equal to δ (see Example 3). The streams are offset relative to each other, i.e., the TSPs of all streams are time-shifted versions of each other. For instance, suppose the desired departure times of packets from one of the stream are $(\tau, \tau+\delta, \tau+2\delta, \ldots)$ for some $\tau \in \mathbb{Z}_+, \delta \in \mathbb{N}$. The desired departure times of packets from another stream traversing the same switch could be $(\tau+\tau', \tau+\tau'+\delta, \tau+\tau'+2\delta, \ldots)$ for some $\tau' \in \mathbb{Z}_+$. Both streams are periodic with IDTs equal to δ, but are offset by τ' slots with respect to each other. We generated the offsets uniformly at random from the set $\{0, 1, \ldots, \delta-1\}$ and varied δ from 18 to 36 slots to vary the load per input port, viz. $16/\delta$. The performance results are depicted in Fig. 9. The efficacy of single subset based operation under uniform switch loading is evident from the plots. For instance, even at $\sim 80\%$ loading of the switch, the average deviation and the variance of the deviation under the MSL-SS policy are -0.3 and 0.2, respectively. This means that on an average, the received service traces for all 256 VOQs deviate from the target stream profiles by only 0.3 time-slots, with a standard deviation of ≈ 0.45 time-slots. Such small negative deviations with small variances can easily be corrected for by allowing traffic streams to gain a lead of 1-2 time-slots (e.g. by using the MSL(ℓ)-SS policy). Roughly speaking, using MSL(ℓ)-SS adds ℓ time-slots to the average deviation, without impacting the variance. For $\ell = 2$, the average deviation for the specific example discussed above would be 1.7. With a standard deviation of 0.44, the probability of missing a deadline (target departure time) will be minimal, even at $\sim 80\%$ loading of the switch.

We also evaluated the performance of the proposed policies under non-uniform periodic loading and Markovian modulated Bernoulli loading (entries of the target stream profile were generated from an MMB process, instead of an i.i.d. Bernoulli process). The performance was observed to be more or less invariant to the statistics of the input traffic streams, underlining the robustness of the proposed policies.

Remark 8: Our simulation results demonstrate that it is possible to render IQ switches nearly *transparent* to deadline sensitive traffic streams by minimizing the distortion of their target profiles. Moreover, this can be

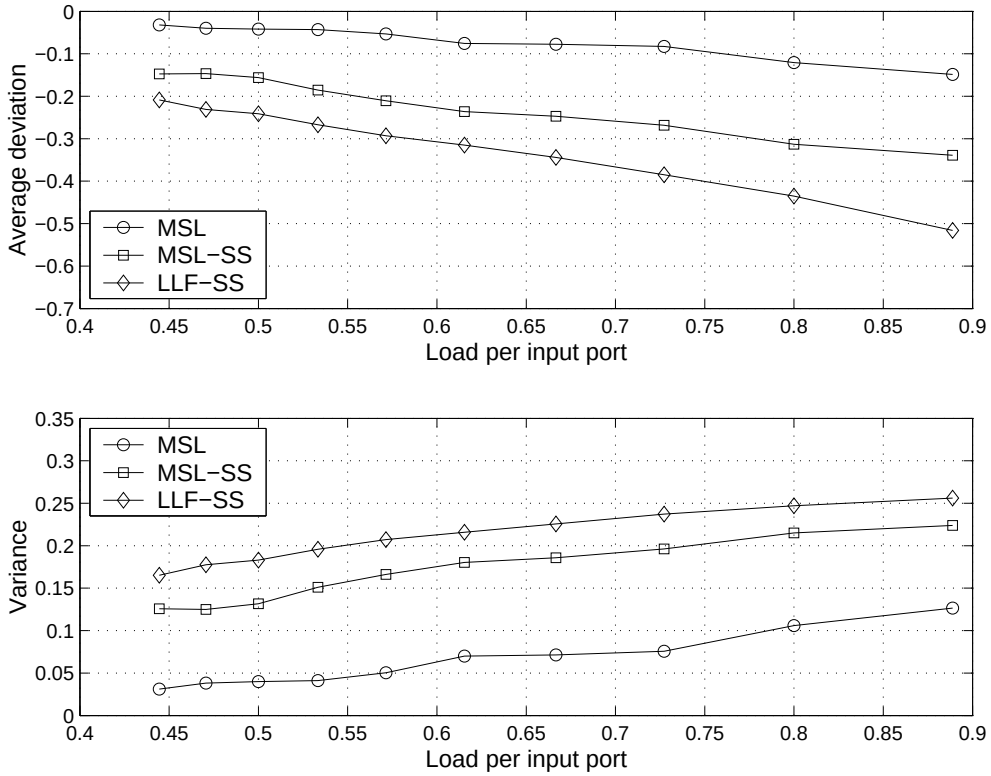

Fig. 9. Performance of MSL, MSL-SS and LLF-SS under uniform periodic loading for an 16×16 switch

accomplished with low complexity online scheduling policies, and with no prior knowledge of the input traffic statistics. For the proposed policies, the transparency of the switch is particularly strong under moderate loading ($< 60\%$), which is a very relevant regime, since a switch is unlikely to be loaded to capacity with real-time traffic. For example, at $\sim 50\%$ loading, for all four loading scenarios simulated here, the average deviation from the target stream profile under all proposed policies is no more -0.3, with a variance below 0.2. Moreover, the proposed policies do not require any prior knowledge of the statistics of the input traffic. For instance, they do not require the streams to be periodic or to be constrained by a leaky bucket, as long as the offered load is within the admissible region of the switch. Finally, the $\mathcal{O}(N^2)$ complexity of the proposed policies render them more amenable to implementation in high performance packet switches vis-à-vis other schedulers proposed in the literature for multiclass periodic traffic, which have a computational complexity of $\mathcal{O}(N^3)$ or $\mathcal{O}(N^4)$, in addition to their high implementation complexity.

VII. Conclusion

We examined the problem of packet switch scheduling for minimizing aggregate distortion of outflow traffic streams with respect to target packet inter-departure times. The study was initially motivated by the need to provide QoS for real-time multimedia traffic over packet networks. The notion of switch configuration subset based control was leveraged to design robust, low complexity, near optimal schedules amenable to implementation in high performance packet switches. Such schedules have been shown to achieve 100% pull-throughput under certain natural statistics of target profiles.

Many theoretical questions remain open, including the pull-throughput region of the switch under general target profile statistics. Moreover, sweeping experimentation is needed to scope out the design and performance of such switches and schedules in broad, diverse target profile regimes.

VIII. Appendix

A. Proof of Theorem 1

Proof: For ease of exposition, we only treat the case where the switch is never idled. A 2×2 switch can be set into two possible configurations, $\mathbf{v}_1 = (1\ 0\ 0\ 1)$ and $\mathbf{v}_2 = (0\ 1\ 1\ 0)$. Given initial state $\mathbf{d}^0$, we say that state $\mathbf{d}$ is

reachable in the t^{th} time-slot if there is a sequence of configurations which drive the switch to state $\mathbf{d}$ in t time-slots. The reachable states in the t^{th} time-slot constitute the set $\mathcal{R}^t = \{\mathbf{d}_k^t \triangleq \mathbf{d}^0 + k\mathbf{v}_1 + (t-k)\mathbf{v}_2 - \mathbf{X}^t\}_{k=0}^t$. The state $\mathbf{d}_k^t$ is reached if the STC selects configuration $\mathbf{v}_1$ in k of the first t time-slots and $\mathbf{v}_2$ in remaining $t-k$ time-slots. The states reachable in the $(t+1)^{st}$ time-slot given state $\mathbf{d}_k^t$ in the t^{th} time-slot are $\mathbf{d}_k^t + \mathbf{v}_1 - \mathbf{x}^{t+1} = \mathbf{d}_{k+1}^{t+1}$ and $\mathbf{d}_k^t + \mathbf{v}_2 - \mathbf{x}^{t+1} = \mathbf{d}_k^{t+1}$. Equivalently, we can identify the state in the t^{th} time-slot by the index k, which increases by 1 in the next time-slot if $\mathbf{v}_1$ is chosen and remains unaltered if $\mathbf{v}_2$ is chosen.

Observe that $\mathbf{d}_k^t$ is a sum of two components:

1) $\mathbf{d}^0 + k\mathbf{v}_1 + (t-k)\mathbf{v}_2$, the evolution of which is determined by the service trace control policy
2) $-\mathbf{X}^t$, the evolution of which is determined by the inter-departure time targets of the input streams.

The first component can be represented using a *directed acyclic graph* (DAG). The root of the DAG is at $\mathbf{d}^0$. Nodes of the DAG at depth t correspond to the policy dependent component of the reachable states in the t^{th} time-slot. There are $t+1$ nodes at depth t, ordered in increasing order of index k from right to left (Fig. 10). The optimal policy traverses the least cost path from the root to one of the leaves.

We use $C(\mathbf{d}_k^t) = i$ to denote that $\Pi_T^\star$ chooses configuration $\mathbf{v}_i$ in the t^{th} time-slot in state corresponding to index k. Also, we define

$$\Omega^t(\mathbf{d}) \triangleq V^t(\mathbf{d}) + \Phi(\mathbf{d}). \tag{37}$$

$$\gamma^t(\mathbf{d}) \triangleq \Omega^{t+1}(\mathbf{d} + \mathbf{v}_1 - \mathbf{x}^t) - \Omega^{t+1}(\mathbf{d} + \mathbf{v}_2 - \mathbf{x}^t). \tag{38}$$

The quantity $\gamma^t(\mathbf{d})$ is interpreted as the *decision function* in state $\mathbf{d}$ in the t^{th} time-slot, i.e., $C(\mathbf{d}^t) = 1$ (configuration $\mathbf{v}_1$ selected) if $\gamma^t(\mathbf{d}) \leq 0$, and $C(\mathbf{d}^t) = 2$ (configuration $\mathbf{v}_2$ selected) otherwise.

We need four auxiliary results to prove the optimality of the myopic policy. The proofs of the auxiliary results are omitted due to space constraints.

1) Auxiliary result 1 (AR1): For any state $\mathbf{d}$, $\Phi(\mathbf{d} + \mathbf{v}_1 - \mathbf{v}_2) - \Phi(\mathbf{d}) \geq \Phi(\mathbf{d}) - \Phi(\mathbf{d} - \mathbf{v}_1 + \mathbf{v}_2)$.

Proof: Recall from Section II-B that the cost functions $\phi_i(\cdot)$ associated with the VOQs are convex. Thus, for the i^{th} VOQ with deviation d_i, the following holds - $\phi_i(d_i+1) - \phi_i(d_i) \geq \phi_i(d_i) - \phi_i(d_i-1)$. Further, from 8, the cost function Φ is the sum of cost functions of all VOQs. Combining the convexity of ϕ_i with the definition of Φ, we arrive at the desired result.

2) Auxiliary result 2 (AR2): For $t = 1, \ldots, T$ and any state $\mathbf{d}$, $V^t(\mathbf{d}+\mathbf{v}_1-\mathbf{v}_2) - V^t(\mathbf{d}) \geq V^t(\mathbf{d}) - V^t(\mathbf{d}-\mathbf{v}_1+\mathbf{v}_2)$.

Proof: The proof is based on inductive arguments. *Base case*: For the base case, $t = T$, it follows from (11) and the boundary conditions $V^{T+1}(\mathbf{d}) = 0$ that

$$V^T(\mathbf{d}) = \min\left\{\Phi(\mathbf{d} + \mathbf{v}_1 - \mathbf{x}^T), \Phi(\mathbf{d} + \mathbf{v}_1 - \mathbf{x}^T)\right\}. \tag{39}$$

Suppose the result is true for some $t < T$, i.e.,

$$V^t(\mathbf{d}' + \mathbf{v}_1 - \mathbf{v}_2) - V^t(\mathbf{d}') \geq V^t(\mathbf{d}') - V^t(\mathbf{d}' - \mathbf{v}_1 + \mathbf{v}_2) \; \forall \; \mathbf{d}'. \tag{40}$$

We will show that the (40) implies that the result is true for $t+1$, i.e.,

$$V^{t+1}(\mathbf{d}' + \mathbf{v}_1 - \mathbf{v}_2) - V^{t+1}(\mathbf{d}') \geq V^{t+1}(\mathbf{d}') - V^{t+1}(\mathbf{d}' - \mathbf{v}_1 + \mathbf{v}_2) \; \forall \; \mathbf{d}'. \tag{41}$$

Setting $\mathbf{d}' = \mathbf{d} + \mathbf{v}_1 - \mathbf{x}^t$ in (40) and invoking AR1, we get

$$\Omega^{t+1}(\mathbf{d} + 2\mathbf{v}_1 - \mathbf{v}_2 - \mathbf{x}^t) - \Omega^{t+1}(\mathbf{d} + \mathbf{v}_1 - \mathbf{x}^t) \geq \Omega^{t+1}(\mathbf{d} + \mathbf{v}_1 - \mathbf{x}^t) - \Omega^{t+1}(\mathbf{d} + \mathbf{v}_2 - \mathbf{x}^t). \tag{42}$$

Similarly, setting $\mathbf{d}' = \mathbf{d} + \mathbf{v}_2 - \mathbf{x}^t$ in (40) and invoking AR1, we get

$$\Omega^{t+1}(\mathbf{d} + \mathbf{v}_1 - \mathbf{x}^t) - \Omega^{t+1}(\mathbf{d} + \mathbf{v}_2 - \mathbf{x}^t) \geq \Omega^{t+1}(\mathbf{d} + \mathbf{v}_2 - \mathbf{x}^t) - \Omega^{t+1}(\mathbf{d} + \mathbf{v}_1 - \mathbf{x}^t). \tag{43}$$

It follows from the definition of $\gamma^t(\cdot)$, (42), and (43) that

$$\gamma^t(\mathbf{d} + \mathbf{v}_1 - \mathbf{v}_2) \geq \gamma^t(\mathbf{d}) \geq \gamma^t(\mathbf{d} - \mathbf{v}_1 + \mathbf{v}_2). \tag{44}$$

In view of (44), four distinct cases arise:

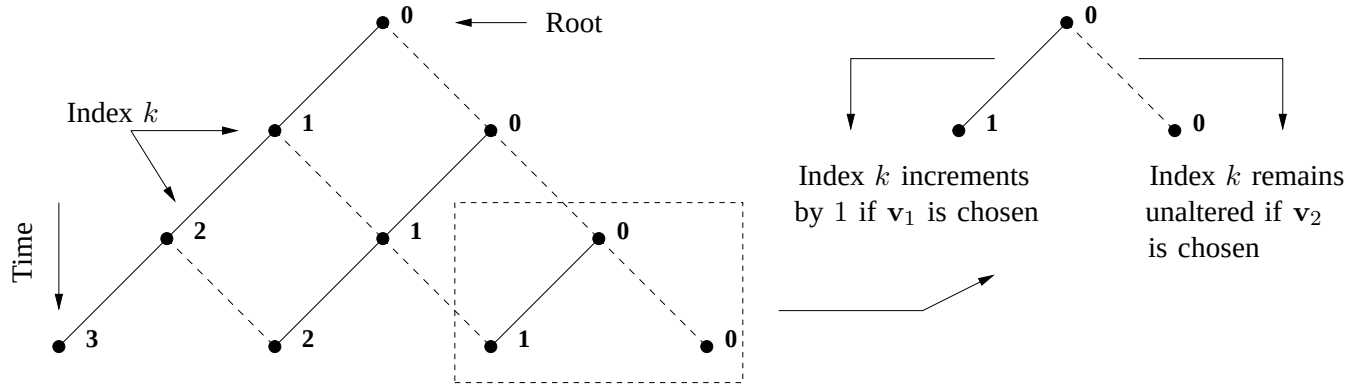

Fig. 10. A DAG based representation of the reachable states in successive time-slots, for a 2×2 switch. The solid lines represent choice of configuration $\mathbf{v}_1$, while the dotted lines represent choice of configuration $\mathbf{v}_2$. The inset depicts the evolution of state index k, based on the choice of configuration in a state.

- $0 \geq \gamma^t(\mathbf{d}+\mathbf{v}_1-\mathbf{v}_2) \geq \gamma^t(\mathbf{d}) \geq \gamma^t(\mathbf{d}-\mathbf{v}_1+\mathbf{v}_2)$: In this case, $C^t(\mathbf{d}-\mathbf{v}_1+\mathbf{v}_2) = C^t(\mathbf{s}) = C^t(\mathbf{d}+\mathbf{v}_1-\mathbf{v}_2) = 1$. We have,

$$\begin{aligned} V^t(\mathbf{d}+\mathbf{v}_1-\mathbf{v}_2) &= \Omega^{t+1}(\mathbf{d}+2\mathbf{v}_1-\mathbf{v}_2-x^t) \\ V^t(\mathbf{d}) &= \Omega^{t+1}(\mathbf{d}+\mathbf{v}_1-\mathbf{x}^t) \\ V^t(\mathbf{d}-\mathbf{v}_1+\mathbf{v}_2) &= \Omega^{t+1}(\mathbf{d}+\mathbf{v}_2-\mathbf{x}^t) \end{aligned}$$

  The result in (41) now follows from the (42) and the above set of equalities.
- $\gamma^t(\mathbf{d}+\mathbf{v}_1-\mathbf{v}_2) \geq \gamma^t(\mathbf{d}) \geq \gamma^t(\mathbf{d}-\mathbf{v}_1+\mathbf{v}_2) \geq 0$: In this case, $C^t(\mathbf{d}-\mathbf{v}_1+\mathbf{v}_2) = C^t(\mathbf{s}) = C^t(\mathbf{d}+\mathbf{v}_1-\mathbf{v}_2) = 2$. We have,

$$\begin{aligned} V^t(\mathbf{d}+\mathbf{v}_1-\mathbf{v}_2) &= \Omega^{t+1}(\mathbf{d}+\mathbf{v}_1-x^t) \\ V^t(\mathbf{d}) &= \Omega^{t+1}(\mathbf{d}+\mathbf{v}_2-\mathbf{x}^t) \\ V^t(\mathbf{d}-\mathbf{v}_1+\mathbf{v}_2) &= \Omega^{t+1}(\mathbf{d}-\mathbf{v}_1+2\mathbf{v}_2-\mathbf{x}^t) \end{aligned}$$

  The result in (41) now follows from the (43) and the above set of equalities.
- $\gamma^t(\mathbf{d}+\mathbf{v}_1-\mathbf{v}_2) \geq 0 \geq \gamma^t(\mathbf{d}) \geq \gamma^t(\mathbf{d}-\mathbf{v}_1+\mathbf{v}_2)$: In this case, $C^t(\mathbf{d}-\mathbf{v}_1+\mathbf{v}_2) = C^t(\mathbf{s}) = 1$ and $C^t(\mathbf{d}+\mathbf{v}_1-\mathbf{v}_2) = 2$. We have,

$$\begin{aligned} V^t(\mathbf{d}+\mathbf{v}_1-\mathbf{v}_2) &= \Omega^{t+1}(\mathbf{d}+\mathbf{v}_1-x^t) \\ V^t(\mathbf{d}) &= \Omega^{t+1}(\mathbf{d}+\mathbf{v}_1-\mathbf{x}^t) \\ V^t(\mathbf{d}-\mathbf{v}_1+\mathbf{v}_2) &= \Omega^{t+1}(\mathbf{d}+\mathbf{v}_2-\mathbf{x}^t) \end{aligned}$$

  The result in (41) now follows from the above set of equalities, the definition of $\gamma^t(\cdot)$, and the assumption that $\gamma^t(\mathbf{d}) \leq 0$.
- $\gamma^t(\mathbf{d}+\mathbf{v}_1-\mathbf{v}_2) \geq \gamma^t(\mathbf{d}) \geq 0 \geq \gamma^t(\mathbf{d}-\mathbf{v}_1+\mathbf{v}_2)$: In this case, $C^t(\mathbf{d}-\mathbf{v}_1+\mathbf{v}_2) = 1$ and $C^t(\mathbf{s}) = C^t(\mathbf{d}+\mathbf{v}_1-\mathbf{v}_2) = 2$. We have,

$$\begin{aligned} V^t(\mathbf{d}+\mathbf{v}_1-\mathbf{v}_2) &= \Omega^{t+1}(\mathbf{d}+\mathbf{v}_1-x^t) \\ V^t(\mathbf{d}) &= \Omega^{t+1}(\mathbf{d}+\mathbf{v}_2-\mathbf{x}^t) \\ V^t(\mathbf{d}-\mathbf{v}_1+\mathbf{v}_2) &= \Omega^{t+1}(\mathbf{d}+\mathbf{v}_2-\mathbf{x}^t) \end{aligned}$$

  The result in (41) now follows from the above set of equalities, the definition of $\gamma^t(\cdot)$, and the assumption that $\gamma^t(\mathbf{d}) > 0$.

Since the four cases considered above are mutually exhaustive, the proof is complete.

3) Auxiliary result 3 (AR3): For $t = 1, \ldots, T$, $\exists\ k_t^\star \in \{0, \ldots, t\}$ such that $C(\mathbf{d}_k^t) = 1\ \forall\ k \leq k_t^\star$ and $C(\mathbf{d}_k^t) = 2\ \forall\ k > k_t^\star$.

Proof: Adding the results of AR1 and AR2, and invoking the definition of $\Omega(\cdot)$, we get

$$\Omega^t(\mathbf{d} + \mathbf{v}_1 - \mathbf{v}_2) - \Omega^t(\mathbf{d}) \geq \Omega^t(\mathbf{d}) - \Omega^t(\mathbf{d} - \mathbf{v}_1 + \mathbf{v}_2). \tag{45}$$

Combining (45) with the definition of $\gamma(\cdot)$, it follows that $\gamma^t(\mathbf{d} + \mathbf{v}_1 - \mathbf{v}_2) \geq \gamma^t(\mathbf{d})$. Finally, from the definition of $\mathbf{d}_k^t$, we get $\mathbf{d}_{k+1}^t - d_k^t = \mathbf{v}_1 - \mathbf{v}_2$. This implies that $\gamma^t(\mathbf{d}_{k+1}^t) \geq \gamma^t(\mathbf{d}_k^t)$, i.e., the decision function $\gamma^t(\mathbf{d}_k^t)$ is a non-decreasing function of k. The proof is based on inductive arguments. We skip the details here. Now, recall that the optimal configuration in state $\mathbf{d}$ at time t is completely determined by the sign of $\gamma^t(\mathbf{d})$. For fixed t, $\gamma^t(\mathbf{d}_k^t)$ can change sign at most once as k increases from 0 to t (by virtue of its monotonicity). In other words, $\exists\ k_t^\star \in \{0, 1, \ldots, t\}$ such that $\gamma^t(\mathbf{d}_k^t) \leq 0\ \forall\ k \leq k_t^\star$ (implying $C(\mathbf{d}_k^t) = 1$) and $\gamma^t(\mathbf{d}_k^t) > 0\ \forall\ k > k_t^\star$ (implying $C(\mathbf{d}_k^t) = 2$).

4) Auxiliary result 4 (AR4): $\arg\min_{k=0,\ldots,t} \Omega^t(\mathbf{d}_k^t) = k_t^\star$.

Proof: We first show that $\arg\min_{k=0,\ldots,t} \Omega^t(\mathbf{d}_k^t) = k_t^\star$. The desired result of AR4 then follows directly. It follows from AR2 that

$$V^t(\mathbf{d}_{k+1}^t) - V^t(\mathbf{d}_k^t) \geq V^t(\mathbf{d}_k^t) - V^t(\mathbf{d}_{k-1}^t). \tag{46}$$

By definition of $k_t^\star$, $C(\mathbf{d}_{k_t^\star+1}^t) = 2$ and $C(\mathbf{d}_{k_t^\star}^t) = 1$. Thus, $V^t(\mathbf{d}_{k_t^\star+1}^t) = V^t(\mathbf{d}_{k_t^\star}^t) = \Omega^{t+1}(\mathbf{d}_{k_{t+1}^\star}^{t+1})$. Setting $k = k_t^\star$ and then $k = k_t^\star + 1$ in (46) we get $V^t(\mathbf{d}_{k_t^\star+2}^t) \geq V^t(\mathbf{d}_{k_t^\star+1}^t) = V^t(\mathbf{d}_{k_t^\star}^t) \geq V^t(\mathbf{d}_{k_t^\star-1}^t)$. Inductively, we conclude that $V^t(\mathbf{d}_{k_t^\star}^t)$ is non-increasing for $k \leq k_t^\star$ and non-decreasing for $k > k_t^\star$, as desired.

Equipped with our auxiliary results, we will show that

$$\begin{aligned} \arg\min\{\Omega^{t+1}(\mathbf{d} + \mathbf{v}_1 - \mathbf{x}^t),\ \Omega^{t+1}(\mathbf{d} + \mathbf{v}_2 - \mathbf{x}^t)\} = \\ \arg\min\{\Phi(\mathbf{d} + \mathbf{v}_1 - \mathbf{x}^t),\ \Phi(\mathbf{d} + \mathbf{v}_2 - \mathbf{x}^t)\}, \end{aligned} \tag{47}$$

which implies that the myopic policy is optimal, because the left side of (47) is the decision of $\Pi_T^\star$ while the right side is the decision of the myopic policy in state $\mathbf{d}$ in the t^{th} time-slot.

Say the switch is in state $\mathbf{d}$ in the t^{th} time-slot and $C(\mathbf{d}) = 2$. The states reachable from $\mathbf{d}$ in the $(t+1)^{st}$ time-slot are $\mathbf{d}_1 = \mathbf{d} + \mathbf{v}_1 - \mathbf{x}^t$ and $\mathbf{d}_2 = \mathbf{d} + \mathbf{v}_2 - \mathbf{x}^t$. Four cases arise, depending on whether $C(\mathbf{d}_1)$ and $C(\mathbf{d}_2)$ are 1 or 2.

5) $C(\mathbf{d}_1) = 2, C(\mathbf{d}_2) = 1$: Since $V^{T+1}(\mathbf{d}) = 0\ \forall\ \mathbf{d}$, the result is trivially true for $t = T$. Let us consider $t < T$. It follows by definition that $V^{t+1}(\mathbf{d}_1) = \Omega^{t+2}(\mathbf{d}_1 + \mathbf{v}_2 - \mathbf{x}^{t+1})$ and $V^{t+1}(\mathbf{d}_2) = \Omega^{t+2}(\mathbf{d}_2 + \mathbf{v}_1 - \mathbf{x}^{t+1})$. However, $\mathbf{d}_1 + \mathbf{v}_2 - \mathbf{x}^{t+1} = \mathbf{d}_2 + \mathbf{v}_1 - \mathbf{x}^{t+1} = \mathbf{d} + \mathbf{v}_1 + \mathbf{v}_2 - \mathbf{x}^t - \mathbf{x}^{t+1}$. Thus, $\Omega^{t+1}(\mathbf{d}_1) - \Omega^{t+1}(\mathbf{d}_2) = \Phi(\mathbf{d}_1) - \Phi(\mathbf{d}_2)$, implying (47).

6) $C(\mathbf{d}_1) = 2, C(\mathbf{d}_2) = 2$: Again the result is trivially true for $t = T$. Let us consider $t < T$. Several possibilities can arise. Since $C(\mathbf{d}_2) = 2$, the state in the $(t+2)^{nd}$ time-slot is $\mathbf{d}_2 + \mathbf{v}_2 - \mathbf{x}^{t+1}$. The next state is determined by the optimal choice of configuration in state $\mathbf{d}_2 + \mathbf{v}_2 - \mathbf{x}^{t+1}$ in the $(t+2)^{nd}$ time-slot, and so on. In general, we can construct a *chain* of states which the switch visits under $\Pi_T^\star$, starting in state $\mathbf{d}_2$ in the $(t+1)^{st}$ time-slot. The chain terminates for one of the following two reasons:

1) The end of the time horizon T is reached.
2) A state is reached where the optimal choice is $\mathbf{v}_1$.

For all states constituting the chain except possibly the last, the optimal configuration is $\mathbf{v}_2$. The optimal configuration in state $\mathbf{d}_1$ in the $(t+1)^{st}$ time-slot is also $\mathbf{v}_2$. Thus, we can construct a similar chain of states originating at $\mathbf{d}_1$, which terminates for one of the two reasons cited above. The chain originating in state $\mathbf{d}_1$ comprises of the states of the type $\mathbf{t}_1^j = \mathbf{d}_1 + j\mathbf{v}_2 + \mathbf{X}^t - \mathbf{X}^{t+j}$ ($j = 0, 1, \ldots$) and the chain originating in state $\mathbf{d}_2$ comprises of the states of the type $\mathbf{t}_2^j = \mathbf{d}_2 + j\mathbf{v}_2 + \mathbf{X}^t - \mathbf{X}^{t+j}$ ($j = 0, 1, \ldots$).

AR3 implies that the chain originating in state $\mathbf{d}_1$ cannot terminate before the chain originating in state $\mathbf{d}_2$ due to the reason 2. AR1 implies $\Phi(\mathbf{t}_1^j) - \Phi(\mathbf{t}_2^j) \leq \Phi(\mathbf{d}_1) - \Phi(\mathbf{d}_2) \triangleq \delta_\Phi$. If both chains terminate due to reason (1), we can show $0 \leq \Omega^{t+1}(\mathbf{d}_1) - \Omega^{t+1}(\mathbf{d}_2) \leq (T - t + 1)\delta_\Phi$, thereby implying (47). Now, suppose the chain originating in state $\mathbf{d}_2$ terminates in the τ^{th} time-slot ($\tau < T$) due to reason (1). We have two further sub-cases: (i) $C(\mathbf{t}_1^{\tau-t}) = 2$ and (ii) $C(\mathbf{t}_1^{\tau-t}) = 1$. For sub-case (i), we can show $0 \leq \Omega^{t+1}(\mathbf{d}_1) - \Omega^{t+1}(\mathbf{d}_2) \leq (\tau - t + 1)\delta_\Phi$, thereby implying (47). Sub-case (ii) cannot arise because we reach a contradiction by virtue of AR4.

7) $C(\mathbf{d}_1) = 1, C(\mathbf{d}_2) = 2$*:* This case violates AR3 and therefore cannot arise.

8) $C(\mathbf{d}_1) = 1, C(\mathbf{d}_2) = 1$*:* This case leads to a contradiction similar to the one obtained in sub-case (ii) of case (2) and therefore cannot arise.

By considering a set of collectively exhaustive cases we have shown that (47) holds when $\mathcal{C}(\mathbf{d}) = 2$. Analogous arguments can be constructed for the case $\mathcal{C}(\mathbf{d}) = 1$. It follows that the optimal finite horizon policy for $N = 2$ is myopic. ■

B. Proof of Lemma 1

Proof: The proof is by induction. We will establish monotonicity of γ_{ij} as a function of n_i. The proof for monotonicity of γ_{ij} as a function of n_j follows similarly.

1) Base Case ($t = T$): By definition of γ_{ij}^T and our choice of boundary conditions, $\gamma_{ij}^T(\mathbf{n} + \mathbf{e}_1) - \gamma_{ij}^T(\mathbf{n}) = \psi_i(n_i + 2 - \widetilde{X}_i^T) - 2\psi_i(n_i + 1 - \widetilde{X}_i^T)] + \psi_i(n_i - \widetilde{X}_i^T) \geq 0$, where the inequality follows from the convexity of ψ_i.

2) Inductive Step: Now, assume that $\gamma_{ij}^{t+1}(\mathbf{n} + \mathbf{e}_i) \geq \gamma_{ij}^{t+1}(\mathbf{n})$ for some $t < T$ and $i \neq j$. We will establish that this assumption implies $\gamma_{ij}^t(\mathbf{n} + \mathbf{e}_i) \geq \gamma_{ij}^t(\mathbf{n})$. By definition,

$$\begin{aligned} \gamma_{ij}^t(\mathbf{n} + \mathbf{e}_i) &= \Omega^{t+1}(\mathbf{n} + 2\mathbf{e}_i) - \Omega^{t+1}(\mathbf{n} + \mathbf{e}_i + \mathbf{e}_j) \\ \gamma_{ij}^t(\mathbf{n}) &= \Omega^{t+1}(\mathbf{n} + \mathbf{e}_i) - \Omega^{t+1}(\mathbf{n} + \mathbf{e}_j). \end{aligned} \tag{48}$$

Several cases arise, depending on the optimal decision in states $\mathbf{n} + 2\mathbf{e}_i$, $\mathbf{n} + \mathbf{e}_i + \mathbf{e}_j$, $\mathbf{n} + \mathbf{e}_i$ and $\mathbf{n} + \mathbf{e}_j$ in the $(t+1)^{st}$ time-slot. Our inductive assumptions imply that the (n_i^{t+1}, n_j^{t+1}) plane gets partitioned into $N+1$ distinct connected decision regions by the optimal policy $\Pi_T^\star$. Consequently, as n_i^{t+1} increases for fixed $\{n_j^{t+1}, j \neq i\}$, $\Pi_T^\star$ *switches over* from $\mathcal{M}_i$ to $\mathcal{M}_k$ for some $k \neq i$. Thereafter, $\Pi_T^\star$ never switches back to $\mathcal{M}_i$. This greatly restricts the number of possible cases we need to consider. We will illustrate a representative case where all four states are in the *interior* of the decision region corresponding to $\mathcal{M}_k$. All cases in which one or more of states of interest occur at the *boundary* of two decision regions can be treated as a combination of the representative cases. It follows from (48),

$$\begin{aligned} \Omega^{t+1}(\mathbf{n} + 2\mathbf{e}_i) &= \Omega^{t+2}(\mathbf{n} + 2\mathbf{e}_i + \mathbf{e}_k) + \Psi(\mathbf{n} + 2\mathbf{e}_i - \widetilde{\mathbf{X}}^{t+1}) \\ \Omega^{t+1}(\mathbf{n} + \mathbf{e}_i + \mathbf{e}_j) &= \Omega^{t+2}(\mathbf{n} + \mathbf{e}_i + \mathbf{e}_j + \mathbf{e}_k) + \Psi(\mathbf{n} + \mathbf{e}_i + \mathbf{e}_j - \widetilde{\mathbf{X}}^{t+1}) \\ \Omega^{t+1}(\mathbf{n} + \mathbf{e}_i) &= \Omega^{t+2}(\mathbf{n} + \mathbf{e}_i + \mathbf{e}_k) + \Psi(\mathbf{n} + \mathbf{e}_i - \widetilde{\mathbf{X}}^{t}) \\ \Omega^{t+1}(\mathbf{n} + \mathbf{e}_j) &= \Omega^{t+2}(\mathbf{n} + \mathbf{e}_j + \mathbf{e}_k) + \Psi(\mathbf{n} + \mathbf{e}_j - \widetilde{\mathbf{X}}^{t}) \end{aligned}$$

It follows that

$$\begin{aligned} \gamma_{ij}^t(\mathbf{n} + \mathbf{e}_i) &= \gamma_{ij}^{t+1}(\mathbf{n} + \mathbf{e}_i + \mathbf{e}_k) + \Psi(\mathbf{n} + 2\mathbf{e}_i - \widetilde{\mathbf{X}}^{t+1}) - \Psi(\mathbf{n} + \mathbf{e}_i + \mathbf{e}_j - \widetilde{\mathbf{X}}^{t+1}) \\ \gamma_{ij}^t(\mathbf{n}) &= \gamma_{ij}^{t+1}(\mathbf{n} + \mathbf{e}_k) + \Psi(\mathbf{n} + \mathbf{e}_i - \widetilde{\mathbf{X}}^{t}) - \Psi(\mathbf{n} + \mathbf{e}_j - \widetilde{\mathbf{X}}^{t}) \end{aligned} \tag{49}$$

$$\begin{aligned} gamma_{ij}^t(\mathbf{n} + \mathbf{e}_i) &= \gamma_{ij}^{t+1}(\mathbf{n} + \mathbf{e}_i + \mathbf{e}_k) + \Psi(\mathbf{n} + 2\mathbf{e}_i - \widetilde{\mathbf{X}}^{t+1}) \\ &\quad - \Psi(\mathbf{n} + \mathbf{e}_i + \mathbf{e}_j - \widetilde{\mathbf{X}}^{t+1}) \\ \gamma_{ij}^t(\mathbf{n}) &= \gamma_{ij}^{t+1}(\mathbf{n} + \mathbf{e}_k) + \Psi(\mathbf{n} + \mathbf{e}_i - \widetilde{\mathbf{X}}^{t}) \\ &\quad - \Psi(\mathbf{n} + \mathbf{e}_j - \widetilde{\mathbf{X}}^{t}). \end{aligned} \tag{50}$$

Also, we have from the base case,

$$\gamma_{ij}^T(\mathbf{n} + \mathbf{e}_i) = \Psi(\mathbf{n} + 2\mathbf{e}_i - \widetilde{\mathbf{X}}^{t+1}) - \Psi(\mathbf{n} + \mathbf{e}_i + \mathbf{e}_j - \widetilde{\mathbf{X}}^{t+1}) \tag{51}$$

$$\gamma_{ij}^T(\mathbf{n}) = \Psi(\mathbf{n} + \mathbf{e}_i - \widetilde{\mathbf{X}}^{t}) - \Psi(\mathbf{n} + \mathbf{e}_j - \widetilde{\mathbf{X}}^{t}) \tag{52}$$

Combining, we get

$$\begin{aligned} \gamma_{ij}^t(\mathbf{n} + \mathbf{e}_i) - \gamma_{ij}^t(\mathbf{n}) &= \underbrace{\gamma_{ij}^{t+1}(\mathbf{n} + \mathbf{e}_i + \mathbf{e}_k) - \gamma_{ij}^{t+1}(\mathbf{n} + \mathbf{e}_k)}_{\geq 0 \text{ by our inductive assumptions}} \\ &\quad + \underbrace{\gamma_{ij}^T(\mathbf{n} + \mathbf{e}_i) - \gamma_{ij}^T(\mathbf{n})}_{\geq 0 \text{ by our base case}} \geq 0, \end{aligned}$$

as desired. ■

C. Proof of Theorem 2

Proof: Define the time and state dependent quadratic Lyapunov function $\mathcal{L}^t(\mathbf{d}^t) \triangleq \langle \mathbf{d}^t, \mathbf{d}^t \rangle$ in state $\mathbf{d}^t$ in the t^{th} time-slot. Given $\mathbf{d}^t = \mathbf{d}$, let $\mathbf{v}^\star = \arg\min_{\mathbf{v}\in\mathcal{V}}\{\langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}\rangle\}$. MSL($\ell$) extracts a partial configuration $\bar{\mathbf{v}}^\star$ from $\mathbf{v}^\star$ by idling all VOQs with deviation ℓ or more.

Given that MSL(ℓ) selects partial configuration $\bar{\mathbf{v}}^\star$ in the t^{th} time-slot in state $\mathbf{d}$, the deviation vector in the $(t+1)^{st}$ time-slot is $\mathbf{d}^{t+1} = \mathbf{d}^t - \mathbf{x}^t + \bar{\mathbf{v}}^\star$. Now define the conditional one step expected drift of the Lyapunov function by

$$\delta^t_{\mathcal{L}}(\mathbf{d}) \triangleq \mathbb{E}\left[\mathcal{L}^{t+1}(\mathbf{d}^{t+1}) - \mathcal{L}^t(\mathbf{d}^t) | \mathbf{d}^t = \mathbf{d}\right]. \tag{53}$$

It follows by definition that

$$\mathcal{L}^{t+1}(\mathbf{d}^{t+1}) = \mathcal{L}^t(\mathbf{d}^t) + 2\langle \mathbf{d}^t, \bar{\mathbf{v}}^\star - \mathbf{x}^t\rangle + \langle \bar{\mathbf{v}}^\star - \mathbf{x}^t, \bar{\mathbf{v}}^\star - \mathbf{x}^t\rangle. \tag{54}$$

Conditioning on $\{\mathbf{d}^t = \mathbf{d}\}$ and taking expectation on both sides of (54) we get

$$\delta^t_{\mathcal{L}}(\mathbf{d}) = 2\langle \mathbf{d}, \bar{\mathbf{v}}^\star\rangle - 2\mathbb{E}[\langle \mathbf{d}, \mathbf{x}^t\rangle] + \langle \bar{\mathbf{v}}^\star, \bar{\mathbf{v}}^\star\rangle - 2\mathbb{E}[\langle \bar{\mathbf{v}}^\star, \mathbf{x}^t\rangle] + \mathbb{E}[\langle \mathbf{x}^t, \mathbf{x}^t\rangle].$$

From the linearity of expectation and the definition of the inner product operator,

$$\begin{aligned}\delta^t_{\mathcal{L}}(\mathbf{d}) = 2\langle \mathbf{d}, \bar{\mathbf{v}}^\star\rangle - 2\sum_{j=1}^{N^2} d_j \mathbb{E}[x^t_j] + \langle \bar{\mathbf{v}}^\star, \bar{\mathbf{v}}^\star\rangle \\ - 2\sum_{j=1}^{N^2} \bar{v}^\star_j \mathbb{E}[x^t_j] + \sum_{j=1}^{N^2} \mathbb{E}[x^t_j x^t_j]\end{aligned} \tag{55}$$

Finally, invoking (33), we have

$$\delta^t_{\mathcal{L}}(\mathbf{d}) = 2\langle \mathbf{d}, \bar{\mathbf{v}}^\star\rangle - 2\langle \mathbf{d}, \boldsymbol{\lambda}\rangle + \langle \bar{\mathbf{v}}^\star, \bar{\mathbf{v}}^\star\rangle - 2\langle \boldsymbol{\lambda}, \bar{\mathbf{v}}^\star\rangle + \langle \boldsymbol{\lambda}, \mathbf{1}\rangle. \tag{56}$$

We will now bound each of the above terms individually.

Note that $\langle \bar{\mathbf{v}}^\star, \bar{\mathbf{v}}^\star\rangle \leq N$, since a configuration vector has no more than N ones. Recall that a complete configuration has exactly N ones, but a partial configuration can have less than N ones, if it idles some of the VOQs. Also, note that $\langle \boldsymbol{\lambda}, \bar{\mathbf{v}}^\star\rangle \geq 0$, since by definition, both the load vector and configuration vector have non-zero entries. Plugging these inequalities into (56), we get

$$\delta^t_{\mathcal{L}}(\mathbf{d}) \leq 2\langle \mathbf{d}, \bar{\mathbf{v}}^\star\rangle - 2\langle \mathbf{d}, \boldsymbol{\lambda}\rangle + N + \langle \boldsymbol{\lambda}, \mathbf{1}\rangle. \tag{57}$$

Consider the BV decomposition of load vector $\boldsymbol{\lambda}$, given by

$$\boldsymbol{\lambda} = \sum_{k=1}^{N!} \alpha_k \mathbf{v}_k, \quad \sum_{k=1}^{N!} \alpha_k = \alpha < 1. \tag{58}$$

It follows that

$$\langle \boldsymbol{\lambda}, \mathbf{1}\rangle = \sum_{k=1}^{N!} \langle \alpha_k \mathbf{v}_k, \mathbf{1}\rangle = \sum_{k=1}^{N!} \alpha_k \langle \mathbf{v}_k, \mathbf{1}\rangle = \sum_{k=1}^{N!} \alpha_k \cdot N = N\alpha, \tag{59}$$

where $\langle \mathbf{v}_k, \mathbf{1}\rangle = 1$ follows because $\mathbf{v}_k$ is a complete configuration. Next, we note that $x^t_i \in \{0, 1\}$, since all entries of the target stream profiles are either 0 or 1. As a result, $\langle \mathbf{d}, \boldsymbol{\lambda}\rangle \geq \langle \mathbf{d} - \mathbf{x}^t, \boldsymbol{\lambda}\rangle$. Substituting for $\boldsymbol{\lambda}$ from (58),

$$\langle \mathbf{d}, \boldsymbol{\lambda}\rangle \geq \langle \mathbf{d} - \mathbf{x}^t, \boldsymbol{\lambda}\rangle = \sum_{k=1}^{N!} \alpha_k \langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}_k\rangle. \tag{60}$$

The definition of the MSL service trace control policy implies

$$\langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}^\star\rangle \leq \langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}\rangle \ \forall\ \mathbf{v} \in \mathcal{V}. \tag{61}$$

Combining (60) and (61),

$$\langle \mathbf{d}, \boldsymbol{\lambda} \rangle \geq \sum_{k=1}^{N!} \alpha_k \langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}^\star \rangle = \alpha \langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}^\star \rangle. \tag{62}$$

Now, summing up both sides of (61) $\forall\ \mathbf{v} \in \mathcal{V}$ and using the fact that $\mathcal{V} = N!$,

$$N! \langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}^\star \rangle \leq \langle \mathbf{d} - \mathbf{x}^t, \sum_{\mathbf{v} \in \mathcal{V}} \mathbf{v} \rangle \tag{63}$$

Since each VOQ is served by exactly $(N-1)!$ complete configurations in $\mathcal{V}$, we have $\sum_{\mathbf{v} \in \mathcal{V}} \mathbf{v} = (N-1)!\mathbf{1}$, implying

$$\langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}^\star \rangle \leq \frac{1}{N} \langle \mathbf{d} - \mathbf{x}^t, \mathbf{1} \rangle. \tag{64}$$

Since all VOQs which are idled under MSL(ℓ) have non-negative updated deviation, $\langle \mathbf{d} - \mathbf{x}^t, \bar{\mathbf{v}}^\star \rangle < \langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}^\star \rangle$. Also, $\langle \mathbf{x}^t, \bar{\mathbf{v}}^\star \rangle \leq N$, since $x_i^t \in \{0, 1\}$ and $\langle \bar{\mathbf{v}}^\star, \mathbf{1} \rangle \leq 1$. We now use the these observations and (62) in (57) to get

$$\begin{aligned}
\delta_{\mathcal{L}}^t(\mathbf{d}) &\leq 2\langle \mathbf{d}, \bar{\mathbf{v}}^\star \rangle - 2\langle \mathbf{d}, \boldsymbol{\lambda} \rangle + N + \langle \boldsymbol{\lambda}, \mathbf{1} \rangle \\
&\leq 2\langle \mathbf{d} - \mathbf{x}^t, \bar{\mathbf{v}}^\star \rangle + 2\langle \mathbf{x}^t, \bar{\mathbf{v}}^\star \rangle - 2\alpha \langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}^\star \rangle + N + N\alpha \\
&\leq 2\langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}^\star \rangle - 2\alpha \langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}^\star \rangle + 3N + N\alpha \\
&\leq 2(1-\alpha)\frac{1}{N} \langle \mathbf{d} - \mathbf{x}^t, \mathbf{1} \rangle + N(3+\alpha) \\
&= 2(1-\alpha)\frac{1}{N} \langle \mathbf{d}, \mathbf{1} \rangle - 2(1-\alpha)\frac{1}{N} \langle \mathbf{x}^t, \mathbf{1} \rangle + N(3+\alpha) \\
&\leq 2(1-\alpha)\frac{1}{N} \langle \mathbf{d}, \mathbf{1} \rangle + N(3+\alpha)
\end{aligned} \tag{65}$$

Before we can proceed further, we need the following lemma.

Lemma 2: If the conditional one-step drift of the quadratic Lyapunov function $\mathcal{L}^t(\mathbf{d})$ satisfies $\delta_{\mathcal{L}}^t(\mathbf{d}^t) \leq \epsilon \langle \mathbf{d}^t, \mathbf{1} \rangle + B,\ \forall\ t > 0, \mathbf{d}^t$ and constants $\epsilon > 0, B > 0$ (independent of state $\mathbf{d}^t$), then

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \sum_{i=1}^{N^2} \mathbb{E}[-d_i^\tau] \leq \frac{B}{\epsilon}.$$

Proof: We have,

$$\delta_{\mathcal{L}}^t(\mathbf{d}) \leq \epsilon \langle \mathbf{d}, \mathbf{1} \rangle + B. \tag{66}$$

Taking expectations on both sides of (66), using the law of iterated expectations, summing up both sides for $\tau = 0, \ldots, T-1$, assuming $\mathbf{d}^0 = \mathbf{0}$, and using $\mathcal{L} \geq 0$, we get

$$\frac{1}{T} \sum_{\tau=0}^{T-1} \langle -\mathbb{E}[\mathbf{d}^\tau], \mathbf{1} \rangle \leq \frac{BT}{\epsilon}. \tag{67}$$

Dividing both sides of (67) by T and taking $\limsup_{T \to \infty}$, it follows

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \sum_{i=1}^{N^2} \mathbb{E}[-d_i^\tau] \leq \frac{B}{\epsilon}.$$

■

As seen from (65), the Lyapunov drift satisfies the condition of Lemma 2 with $\epsilon = 2(1-\alpha)/N > 0$ and $B = N(3+\alpha) > 0$. Since the deviation of any VOQ under the MSL(ℓ) policy is upper bounded by $\ell > 0$, we get from Lemma 2 $\forall\ j$

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \mathbb{E}[-d_j^t] \leq \frac{B}{\epsilon} + (N^2 - 1)\ell < \infty, \tag{68}$$

implying $\liminf_{t \to \infty} \mathbb{E}[d_j^t] > -\infty\ \forall\ j$, as desired. ■

D. Proof of Theorem 3

Proof: It can be shown using Lyapunov methods that LLF guarantees finite lags to all meta-queues in the single server model of Section IV-B if the inverse of the average inter-departure time targets of all meta-queues sum to less than 1. The proof of this result is very similar to the proof of Theorem 2 and is therefore omitted. Bounded lags for all meta-queues in the single server model imply bounded lags for all VOQs in the switch, since the lag of a meta-queue is the maximum of the lag of its constituent VOQs (see Section IV-E). Under uniform i.i.d. loading, i.e., $\boldsymbol{\lambda} = \lambda \mathbf{1}$ for some $\lambda < 1/N$, the average inter-departure time target for every VOQ, and hence for every meta-queue in every configuration subset is N/λ. Consequently, the sum of the inverse of the average meta-queue inter-departure time targets sum to $\lambda < 1$, implying the desired result. ■

E. Proof of Theorem 4

Proof: Consider $\mathcal{L}^t(\mathbf{d}^t)$ and $\delta^t_{\mathcal{L}}(\mathbf{d})$, as defined in the proof of Theorem 2. Assume that the switch operates in configuration subset $\mathcal{S}_{\mathbf{v}} = \{\mathcal{C}^k(\mathbf{v})\}_{k=0}^{N-1}$. Let

$$k^\star = \underset{k=0,\ldots,N-1}{\arg\min} \langle \mathbf{d} - \mathbf{x}^t, \mathcal{C}^k(\mathbf{v}) \rangle. \tag{69}$$

MSL(ℓ)-SS selects a partial configuration $\bar{\mathbf{v}}^\star$ which is extracted from the complete configuration $\mathcal{C}^{k^\star}(\mathbf{v})$. Following the arguments in the proof of Theorem 2, we get from (57)

$$\delta^t_{\mathcal{L}}(\mathbf{d}) \leq 2\langle \mathbf{d}, \bar{\mathbf{v}}^\star \rangle - 2\langle \mathbf{d}, \boldsymbol{\lambda} \rangle + N + \langle \boldsymbol{\lambda}, \mathbf{1} \rangle. \tag{70}$$

As always, we will bound each of these terms individually. Since $\boldsymbol{\lambda} = \lambda \mathbf{1}$ for some $\lambda < \dfrac{1}{N}$ for uniform loading, it follows that $\langle \boldsymbol{\lambda}, \mathbf{1} \rangle = \lambda N^2$. Next, it follows from the definition of MSL-SS that

$$\langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}^\star \rangle \leq \langle \mathbf{d} - \mathbf{x}^t, \mathcal{C}^k(\mathbf{v}) \rangle \; \forall \; k \in \{0, 1, \ldots, N-1\} \tag{71}$$

Summing both sides of the equation over k we get from (17)

$$N\langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}^\star \rangle \leq \langle \mathbf{d} - \mathbf{x}^t, \sum_{k=0}^{N-1} \mathcal{C}^k(\mathbf{v}) \rangle = \langle \mathbf{d} - \mathbf{x}^t, \mathbf{1} \rangle \leq \langle \mathbf{d}, \mathbf{1} \rangle. \tag{72}$$

Using $\boldsymbol{\lambda} = \lambda \mathbf{1}$ once again, we get $\langle \mathbf{d}, \boldsymbol{\lambda} \rangle = \lambda \langle \mathbf{d}, \mathbf{1} \rangle$. Since all VOQs which are idled under the MSL(ℓ)-SS policy have non-negative updated deviation, $\langle \mathbf{d} - \mathbf{x}^t, \bar{\mathbf{v}}^\star \rangle < \langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}^\star \rangle$. Also, $\langle \mathbf{x}^t, \bar{\mathbf{v}}^\star \rangle \leq N$, since $x^t_i \in \{0, 1\}$ and $\langle \bar{\mathbf{v}}^\star, \mathbf{1} \rangle \leq 1$. We now use the these observations and (72) in (70) to get

$$\begin{aligned}
\delta^t_{\mathcal{L}}(\mathbf{d}) &\leq 2\langle \mathbf{d}, \bar{\mathbf{v}}^\star \rangle - 2\langle \mathbf{d}, \boldsymbol{\lambda} \rangle + N + \langle \boldsymbol{\lambda}, \mathbf{1} \rangle \\
&= 2\langle \mathbf{d} - \mathbf{x}^t, \bar{\mathbf{v}}^\star \rangle + 2\langle \mathbf{x}^t, \bar{\mathbf{v}}^\star \rangle - 2\lambda \langle \mathbf{d}, \mathbf{1} \rangle + N + N^2\lambda \\
&\leq 2\langle \mathbf{d} - \mathbf{x}^t, \mathbf{v}^\star \rangle - 2\lambda \langle \mathbf{d}, \mathbf{1} \rangle + 3N + N^2\lambda \\
&\leq \frac{2}{N}\langle \mathbf{d}, \mathbf{1} \rangle - 2\lambda \langle \mathbf{d}, \mathbf{1} \rangle + N(3 + N\lambda) \\
&= 2(1 - N\lambda)\frac{1}{N}\langle \mathbf{d}, \mathbf{1} \rangle + N(3 + N\lambda).
\end{aligned} \tag{73}$$

Thus, we have established $\delta^t_{\mathcal{L}}(\mathbf{d}) \leq \epsilon \langle \mathbf{d}, \mathbf{1} \rangle + B$, for $\epsilon = 2(1 - N\lambda)/N$ and $B = N(3 + N\lambda) > 0$. Since the deviation of any VOQ under the MSL(ℓ)-SS policy is upper bounded by $\ell > 0$, it follows from Lemma 2 that $\forall \; j$

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \mathbb{E}[-d^t_j] \leq \frac{B}{\epsilon} + (N^2 - 1)\ell < \infty, \tag{74}$$

implying $\liminf_{t \to \infty} \mathbb{E}[d^t_j] > -\infty \; \forall \; j$, as desired. ■

F. Proof of Theorem 5

Proof: Define $\delta^t_{\mathcal{L}}(\mathbf{d},k)$ as the expected drift in the Lyapunov function $\mathcal{L}^t(\cdot)$ conditioned on the deviation vector $\mathbf{d}$ *and* the choice of configuration subset $\mathcal{S}_k$ in the t^{th} time-slot. If partial configuration $\bar{\mathbf{v}}^\star(k)$ derived from configuration $\mathcal{C}^{i^\star(k)}(\mathbf{v}_k) \in \mathcal{S}_k$ is chosen in the t^{th} time-slot, $\delta^t_{\mathcal{L}}(\mathbf{d},k)$ is given by (56), with $\bar{\mathbf{v}}^\star$ replaced by $\bar{\mathbf{v}}^\star(k)$. Unconditioning with respect to the choice of configuration subset $\mathcal{S}_k$ yields

$$\begin{aligned}\delta^t_{\mathcal{L}}(\mathbf{d}) &= \sum_{k=1}^{(N-1)!} \theta_k \delta^t_{\mathcal{L}}(\mathbf{d},k)\\ &= -2\langle \mathbf{d},\boldsymbol{\lambda}\rangle - 2\sum_{k=1}^{(N-1)!} \underbrace{\theta_k \langle \mathcal{C}^{i^\star(k)}(\mathbf{v}_k),\boldsymbol{\lambda}\rangle}_{\geq 0}\\ &\quad + 2\underbrace{\sum_{k=1}^{(N-1)!} \theta_k \langle \mathbf{d},\mathcal{C}^{i^\star(k)}(\mathbf{v}_k)\rangle}_{W^\star} + \langle \boldsymbol{\lambda},\mathbf{1}\rangle + N\\ &\leq -2\langle \mathbf{d},\boldsymbol{\lambda}\rangle + 2W^\star + N(1+\alpha).\end{aligned} \tag{75}$$

We will bound the terms $\langle \mathbf{d},\boldsymbol{\lambda}\rangle$ and $W^\star$ to arrive at the desired result. It follows from the definition of the MSL-RS policy that

$$\langle \mathbf{d}-\mathbf{x}^t, \mathcal{C}^{i^\star(k)}(\mathbf{v}_k)\rangle \leq \langle \mathbf{d}-\mathbf{x}^t, \mathcal{C}^{i}(\mathbf{v}_k)\rangle \;\forall\; i \in \{0,1,\ldots,N-1\}. \tag{76}$$

Summing both sides of the equation and invoking (17), we get

$$\langle \mathbf{d}-\mathbf{x}^t, \mathcal{C}^{i^\star(k)}(\mathbf{v}_k)\rangle \leq \frac{1}{N}\langle \mathbf{d},\mathbf{1}\rangle \tag{77}$$

Since $x^t_i \in \{0,1\}$ and a configuration vector has no more than N ones, for any configuration vector $\mathbf{v}$, we have $\langle \mathbf{v},\mathbf{x}^t\rangle \leq N$. Using this and the fact that $\{\theta_k\}$ is a probability distribution (implying $\sum_k \theta_k = 1$), we get

$$\sum_{k=1}^{(N-1)!} \theta_k \langle \mathcal{C}^{i^\star(k)}(\mathbf{v}_k),\mathbf{x}^t\rangle \leq N \leq \sum_{k=1}^{(N-1)!} \theta_k \cdot N = N. \tag{78}$$

From (77) and (78) it follows

$$\langle \mathbf{d},\mathcal{C}^{i^\star(k)}(\mathbf{v}_k)\rangle \leq \frac{1}{N}\langle \mathbf{d},\mathbf{1}\rangle + \langle \mathcal{C}^{i^\star(k)}(\mathbf{v}_k),\mathbf{x}^t\rangle \tag{79}$$

Taking the expectation on both sides of the above equation with respect to the distribution $\{\theta_k\}$ and invoking the definition of $W^\star$, we get

$$W^\star \leq \frac{1}{N}\langle \mathbf{d},\mathbf{1}\rangle + N. \tag{80}$$

Now, consider the BV decomposition of load vector $\boldsymbol{\lambda}$ as given by (35), i.e.,

$$\boldsymbol{\lambda} = \sum_{k=1}^{(N-1)!}\sum_{i=0}^{N-1} \zeta_{ik}\mathcal{C}^i(\mathbf{v}_k), \quad \sum_{k=1}^{(N-1)!}\sum_{i=0}^{N-1} \zeta_{ik} = \zeta < 1. \tag{81}$$

Using $\langle \boldsymbol{\lambda}, \mathbf{x}^t \rangle \geq 0$, the definition of MSL-RS in (76), and the definition of θ_k in (36)

$$\begin{aligned}
\langle \mathbf{d}, \lambda \rangle \geq \langle \mathbf{d} - \mathbf{x}^t, \boldsymbol{\lambda} \rangle &= \sum_{k=1}^{(N-1)!} \sum_{i=0}^{N-1} \zeta_{ik} \langle \mathbf{d} - \mathbf{x}^t, \mathcal{C}^i(\mathbf{v}_k) \rangle \\
&\geq \sum_{k=1}^{(N-1)!} \sum_{i=0}^{N-1} \zeta_{ik} \langle \mathbf{d} - \mathbf{x}^t, \mathcal{C}^{i^\star(k)}(\mathbf{v}_k) \rangle \\
&= \sum_{k=1}^{(N-1)!} \langle \mathbf{d} - \mathbf{x}^t, \mathcal{C}^{i^\star(k)}(\mathbf{v}_k) \rangle \sum_{i=0}^{N-1} \zeta_{ik} \\
&= \zeta \sum_{k=1}^{(N-1)!} \theta_k \langle \mathbf{d} - \mathbf{x}^t, \mathcal{C}^{i^\star(k)}(\mathbf{v}_k) \rangle = \zeta W^\star
\end{aligned} \tag{82}$$

Substituting (80) and (82) into (75) and noting that $\alpha = \zeta$,

$$\begin{aligned}
\delta_{\mathcal{L}}^t(\mathbf{d}) &\leq -2\zeta W^\star + 2W^\star + N(1+\zeta) \\
&= 2(1-\zeta)W^\star + N(1+\zeta) \\
&\leq 2(1-\zeta)\frac{1}{N}\langle \mathbf{d}, \mathbf{1} \rangle + 2N(1-\zeta) + N(1+\zeta) \\
&= 2(1-\zeta)\frac{1}{N}\langle \mathbf{d}, \mathbf{1} \rangle + N(3-\zeta)
\end{aligned} \tag{83}$$

We have shown that $\delta_{\mathcal{L}}^t(\mathbf{d}) \leq \epsilon \langle \mathbf{d}, \mathbf{1} \rangle + B$, for $\epsilon = 2(1-\zeta)/N > 0$ and $B = N(3-\zeta) > 0$. Since the deviation of any VOQ under the MSL(ℓ)-RS policy is upper bounded by $\ell > 0$, it follows from Lemma 2 that $\forall\ j$

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \mathbb{E}[-d_j^t] \leq \frac{B}{\epsilon} + (N^2 - 1)\ell < \infty, \tag{84}$$

implying $\liminf_{t \to \infty} \mathbb{E}[d_j^t] > -\infty\ \forall\ j$, as desired.

■

G. Proof of Theorem 6

Proof: Suppose MSL(ℓ) is used in the t^{th} time-slot, followed by MSL-SS(ℓ) in the $(t+1)^{st}, \ldots, (t+P-1)^{th}$ time-slots, and MSL(ℓ) again in the $(t+P)^{th}$ time-slot. We are interested in computing the $P+1$ step conditional expected drift in the Lyapunov function $\mathcal{L}^t(\mathbf{d}^t)$ of Theorem 2, given by

$$\begin{aligned}
\delta_{\mathcal{L}}^{t,P} &\triangleq \mathbb{E}[\mathcal{L}^{t+P+1}(\mathbf{d}^{t+P+1}) - \mathcal{L}^t(\mathbf{d}^t) | \mathbf{d}^t = \mathbf{d}] \\
&= \sum_{p=0}^{P} \underbrace{\mathbb{E}[\mathcal{L}^{t+p+1}(\mathbf{d}^{t+p+1}) - \mathcal{L}^{t+p}(\mathbf{d}^{t+p}) | \mathbf{d}^t = \mathbf{d}]}_{\mu_p}.
\end{aligned} \tag{85}$$

The key idea is to bound each term μ_p individually and then obtain a bound on their sum of the form (66). For ease of exposition, we illustrate the case $P = 2$. The proof extends in straightforward fashion to $P > 2$ (with more algebra).

Suppose (partial) configuration $\bar{\mathbf{v}}_\tau^\star$ is selected in the τ^{th} time-slot[5]. From Theorem 2, we have

$$\mu_0 \leq 2\langle \mathbf{d}, \bar{\mathbf{v}}_t^\star \rangle - 2\langle \mathbf{d}, \lambda \rangle + N(1+\alpha). \tag{86}$$

From the definition of $\mathcal{L}$ and $\mathbf{d}^{t+1} = \mathbf{d}^t + \bar{\mathbf{v}}_t^\star - \mathbf{x}^t$, it follows

$$\mu_1 \leq 2\langle \mathbf{d}, \bar{\mathbf{v}}_{t+1}^\star \rangle - 2\langle \mathbf{d}, \boldsymbol{\lambda} \rangle + 3N(1+\alpha). \tag{87}$$

[5]So far, we have been suppressing the time dependence of $\bar{\mathbf{v}}^\star$ in all the proofs, since we were only considering one-step drifts.

By definition, MSL(ℓ)-SS is used in the $(t+1)^{st}$ time-slot on the subset generated by (the complete configuration corresponding to) $\bar{\mathbf{v}}_t^\star$. This can be used to show $\langle \mathbf{d}, \bar{\mathbf{v}}_{t+1}^\star - \bar{\mathbf{v}}_t^\star \rangle < 2N\alpha$. It follows that

$$\mu_1 \leq 2\langle \mathbf{d}, \bar{\mathbf{v}}_t^\star \rangle - 2\langle \mathbf{d}, \lambda \rangle + 3N + 5N\alpha. \tag{88}$$

Next, since MSL(ℓ) is used in the $(t+2)^{nd}$ slot, it follows

$$\mu_2 \leq \frac{2}{N}(1-\alpha)\langle \mathbf{d}, \mathbf{1} \rangle + 7N + 11N\alpha, \tag{89}$$

Finally, from Theorem 2, $\langle \mathbf{d}, \bar{\mathbf{v}}_t^\star \rangle - \langle \mathbf{d}, \boldsymbol{\lambda} \rangle \leq \frac{1}{N}(1-\alpha)\langle \mathbf{d}, \mathbf{1} \rangle + N$. Combining the above inequalities, we get

$$\delta_{\mathcal{L}}^{t,2} \leq \underbrace{\frac{6}{N}(1-\alpha)}_{\epsilon}\langle \mathbf{d}, \mathbf{1} \rangle + \underbrace{15N + 17N\alpha}_{B}. \tag{90}$$

The desired result now follows from arguments similar to those presented in the proof of Theorem 2. A similar bound can be established for any $P > 2$, with $\epsilon = 2(P+1)(1-\alpha)/N$ and a constant B which is an increasing function of P.

■

REFERENCES

[1] A.K. Parekh and R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: single node case", *IEEE/ACM Trans. Networking*, vol. 1, pp. 344-357, Jun. 1993.

[2] A.K. Parekh and R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: multiple nodes case", *IEEE/ACM Trans. Networking*, vol. 2, pp. 137-150, Apr. 1994.

[3] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch", *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1260-1267, Aug. 1999.

[4] P. Giaccone, B. Prabhakar, and D. Shah, "Randomized scheduling algorithms for high-aggregate bandwidth switches", *IEEE JSAC*, vol. 21, no. 4, pp. 546-559, May 2003.

[5] D. Shah and M. Kopikare, "Delay bounds for approximate maximum weight matching algorithms for input queued switches", *Proc. IEEE INFOCOM*, pp. 1024-1031, New York, NY, Jun. 2002.

[6] K. Ross and N. Bambos, "Projective Cone Scheduling (PCS) Algorithms for Packet Switches of Maximal Throughput" to appear in *IEEE/ACM Trans. Networking*. Early version appeared as "Local search scheduling algorithms for maximal throughput in packet switches", in Proc. IEEE INFOCOM, 2004, vol. 2, pp. 1158 1169.

[7] T. Inukai, "An efficient SS/TDMA time slot assignment algorithm", *IEEE Trans. Commun.*, vol. 27, no. 10, pp. 1449-1455, Oct. 1979.

[8] I.R. Philip and J.W.S Liu, "SS/TDMA scheduling of real-time periodic messages", *Proc. ICTS*, pp. 244-251, Mar. 1996.

[9] J. Giles and B. Hajek, "Scheduling multirate periodic traffic in a packet switch", *Proc. CISS*, Baltimore, MD, Mar. 1997.

[10] I.A. Rai and M. Alanyali, "Uniform weighted round robin scheduling algorithms for input queued switches", *Proc. IEEE ICC*, pp. 2028-2032, Helsinki, Finland, Jun. 2001.

[11] Y. Lee, J. Lou, J. Luo, X. Shen, "An Efficient Packet Scheduling Algorithm with Deadline Guarantees for Input-Queued Switches", *IEEE/ACM Trans. Networking*, vol. 15, no. 1, pp. 212-225, 2007.

[12] S. Li and N. Ansari, "Input-queued switching with QoS guarantees", *Proc. IEEE INFOCOM*, pp. 1152-1159, New York, NY, Mar. 1999.

[13] C.S. Chang, W.J. Chen, and H.Y. Huang, "Birkhoff-von Neumann input-buffered crossbar switches for guaranteed-rate services", *IEEE Trans. Commun.*, vol. 49, no. 7, pp. 1145-1147, Jul. 2001.

[14] C.S. Chang, D.S. Lee, and C.Y Yue, "Providing guaranteed rate services in the load balanced Birkhoff-von Neumann switches", *Proc. IEEE INFOCOM*, pp. 1622-1632, San Francisco, CA, Apr. 2003.

[15] C.S. Chang, D.S. Lee, and Y.S. Jou, "Load balanced Birkhoff-von Neumann switches: part I: one-stage buffering", *Comp. Commun.*, vol. 25, no. 6, pp. 611-622, Apr. 2002.

[16] I. Keslassy, M. Kodialam, T.V. Lakshman, and D. Stiliadis, "On guaranteed smooth scheduling for input-queued switches", *Proc. IEEE INFOCOM*, pp. 1384-1394, San Francisco, CA, Apr. 2003.

[17] E.D. Jensen, C.D. Locke, and H. Tokuda, "A time-driven scheduling model for real-time systems", *Proc. IEEE RTSS*, pp. 112-122, Dec. 1985.

[18] R.L. Cruz, "Quality of service guarantees in virtual circuit switched networks", *IEEE JSAC*, vol. 13, no. 6, pp. 1048-1056, Aug. 1995.

[19] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms*, 2^{nd} Ed., MIT Press and McGraw Hill, 1991.

[20] A. Dua and N. Bambos, "Low-jitter scheduling algorithms for deadline-aware packet switches", *Proc. IEEE Globecom*, San Francisco, CA, Dec. 2006.

[21] A. Dua and N. Bambos, "Scheduling with soft deadlines for input queued packet switches", *Proc. 44^{th} Allerton Conference on Commun., Comp., and Contr.*, Allerton, IL, Sep. 2006.

[22] D. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1 & 2, 2^{nd} Ed., Athena Scientific, 2000.

[23] J. Walrand, *An Introduction to Queueing Networks*, Englewood Cliffs, NJ: Prentice Hall, 1988.

[24] M.J. Neely, E. Modiano, and C.E. Rohrs, “Dynamic power allocation and routing for time-varying wireless networks”, *IEEE JSAC*, vol. 23, no. 1, pp. 89-103, Jan. 2005.

[25] P.R. Kumar and S.P. Meyn, “Stability of queuing networks and scheduling policies”, *IEEE Trans. on Automat. Contr.*, vol. 40, no. 2, pp. 251-260, Feb. 1995.